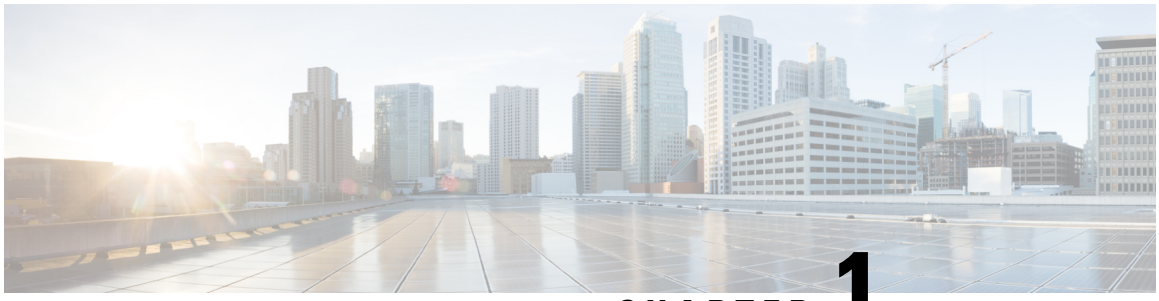
Quality of Service Configuration Guide, Cisco IOS XE Cupertino 17.8.x (Catalyst 9600 Switches)

First Published: 2022-04-09

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Configuring Auto-QoS

Prerequisites for Auto-QoS

The prerequisites for auto-QoS are the same as the prerequisites for standard QoS.

Restrictions for Auto-QoS

The following are restrictions for auto-QoS:

- Auto-qos is not supported on SVI interfaces.
- Do not configure the **auto qos voip cisco-phone** option for IP phones that support video. This option causes DSCP markings of video packets to get overwritten, because these packets do not have Expedited Forwarding priority, which results in these packets getting classified in the class-default class.

Note This restriction does not apply to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2).

- Auto-QoS does not generate configuration when it is pushed from the startup-configuration using the **auto qos voip cisco-phone** command to the running-configuration. This is expected behavior and this is to prevent overwriting of user-created customized QoS policies by the default configuration, if any, every time the command **auto qos voip cisco-phone** is pushed from the startup-config.

  You can use any of the following workarounds for this limitation:

- Configure the **auto qos voip cisco-phone** command manually on the switch interfaces.
- For new switches, if you push auto-QoS commands through startup-config, the command should include each of the following as part of the standard template

  1. Interface-level:
     - **trust device cisco-phone**
     - **auto qos voip cisco-phone**
     - **service-policy input** AutoQos-4.0-CiscoPhone-Input-Policy
     - **service-policy output** AutoQos-4.0-Output-Policy
  2. Global-level:
     - Class-map
     - Policy-map
     - ACL(ACE)

- If the **auto qos voip cisco-phone** command is already configured on an interface but policies are not being generated, disable the command from all the interfaces and reconfigure the command on each interface manually.

Information About Configuring Auto-QoS

Auto-QoS Overview

You can use the auto-QoS feature to simplify the deployment of QoS features. Auto-QoS determines the network design and enables QoS configurations so that the switch can prioritize different traffic flows.

The switch employs the MQC model. This means that instead of using certain global configurations, auto-QoS applied to any interface on a switch configures several global class maps and policy maps.

Auto-QoS matches traffic and assigns each matched packet to qos-groups. This allows the output policy map to put specific qos-groups into specific queues, including into the priority queue.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2), the egress queuing policy only maps traffic-class to queues. For ingress policy, traffic-class has to be set specifically, otherwise the default mapping will use only traffic-class 0 or 7.

QoS is needed in both directions, both on inbound and outbound. When inbound, the switch port needs to trust the DSCP in the packet (done by default). When outbound, the switch port needs to give voice packets "front of line" priority. If voice is delayed too long by waiting behind other packets in the outbound queue, the end host drops the packet because it arrives outside of the receive window for that packet.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2), DSCP is trusted by default for routed packets, and CoS is trusted by default for bridged packets.

Auto-QoS Compact Overview

When you enter an auto-QoS command, the switch displays all the generated commands as if the commands were entered from the CLI. You can use the auto-QoS compact feature to hide the auto-QoS generated commands from the running configuration. This would make it easier to comprehend the running-configuration and also help to increase efficient usage of memory.

Auto-QoS Global Configuration Templates

In general, an auto-QoS command generates a series of class maps that either match on ACLs or on DSCP and/or CoS values to differentiate traffic into application classes. An input policy is also generated, which matches the generated classes and in some cases, polices the classes to a set bandwidth. Eight egress-queue class maps are generated. The actual egress output policy assigns a queue to each one of these eight egress-queue class maps.

The auto-QoS commands only generate templates as needed. For example, the first time any new auto-QoS command is used, global configurations that define the eight queue egress service-policy are generated. From this point on, auto-QoS commands applied to other interfaces do not generate templates for egress queuing because all auto-QoS commands rely on the same eight queue models, which have already been generated from the first time a new auto-QoS command was used.

Auto-QoS Policy and Class Maps

After entering the appropriate auto-QoS command, the following actions occur:

- Specific class maps are created.
- Specific policy maps (input and output) are created.
- Policy maps are attached to the specified interface.
- Trust level for the interface is configured.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2), DSCP is trusted by default for routed packets, and CoS is trusted by default for bridged packets.

Effects of Auto-QoS on Running Configuration

When auto-QoS is enabled, the **auto qos** interface configuration commands and the generated global configuration are added to the running configuration.

The switch applies the auto-QoS-generated commands as if the commands were entered from the CLI. An existing user configuration can cause the application of the generated commands to fail or to be overridden by the generated commands. These actions may occur without warning. If all the generated commands are successfully applied, any user-entered configuration that was not overridden remains in the running configuration. Any user-entered configuration that was overridden can be retrieved by reloading the switch without saving the current configuration to memory. If the generated commands are not applied, the previous running configuration is restored.

Effects of Auto-Qos Compact on Running Configuration

If auto-QoS compact is enabled:

- Only the auto-QoS commands entered from the CLI are displayed in running-config.
- The generated global and interface configurations are hidden.
- When you save the configuration, only the auto-qos commands you have entered are saved (and not the hidden configuration).
- When you reload the switch, the system detects and re-executes the saved auto-QoS commands and the AutoQoS SRND4.0 compliant config-set is generated.

Note Do not make changes to the auto-QoS-generated commands when auto-QoS compact is enabled, because user-modifications are overridden when the switch reloads.

When auto-qos global compact is enabled:

- **show derived-config** command can be used to view hidden Auto-QoS global Compact derived commands.
- Auto-QoS global Compact commands will not be stored to memory. They will be regenerated every time the switch is reloaded.
- When compaction is enabled, auto-qos generated commands should not be modified.
- If the interface is configured with auto-QoS and if Auto-QoS global Compact needs to be disabled, auto-QoS should be disabled at interface level first.

How to configure Auto-QoS

Configuring Auto-QoS

For optimum QoS performance, configure auto-QoS on all the devices in your network.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE 1/0/1`	Specifies the port that is connected to a VoIP port, video device, or the uplink port that is connected to another trusted switch or router in the network interior, and enters the interface configuration mode.
Step 3	Depending on your auto-QoS configuration, use one of the following commands: • **auto qos voip** {**cisco-phone** | **cisco-softphone** | **trust**} • **auto qos video** {**cts** | **ip-camera** | **media-player**} • **auto qos classify** [**police**] • **auto qos trust** {**cos** | **dscp**} **Example:** `Device(config-if)# auto qos trust dscp`	The following commands enable auto-QoS for VoIP: • **auto qos voip cisco-phone**: If the port is connected to a Cisco IP Phone, the QoS labels of incoming packets are only trusted (conditional trust through CDP) when the telephone is detected. **Note** Do not configure the **auto qos voip cisco-phone** option for IP phones that support video. This option causes DSCP markings of video packets to get overwritten, because these packets do not have Expedited Forwarding priority, which results in these packets getting classified in the class-default class. • **auto qos voip cisco-softphone**: The port is connected to device running the Cisco SoftPhone feature. This command generates a QoS configuration for interfaces connected to PCs running the Cisco IP SoftPhone application and mark, as well as police traffic coming from such interfaces. Ports configured with this command are considered untrusted. • **auto qos voip trust**: The uplink port is connected to a trusted switch or router, and the VoIP traffic classification in the ingress packet is trusted.

	Command or Action	Purpose
		The following commands enable auto-QoS for the specified video device (system, camera, or media player): • **auto qos video cts**: A port connected to a Cisco Telepresence system. QoS labels of incoming packets are only trusted (conditional trust through CDP) when a Cisco TelePresence is detected. • **auto qos video ip-camera**: A port connected to a Cisco video surveillance camera. QoS labels of incoming packets are only trusted (conditional trust through CDP) when a Cisco camera is detected. • **auto qos video media-player**: A port connected to a CDP-capable Cisco digital media player. QoS labels of incoming packets are only trusted (conditional trust through CDP) when a digital media player is detected. The following command enables auto-QoS for classification: • **auto qos classify police**: This command generates a QoS configuration for untrusted interfaces. The configuration places a service-policy on the interface to classify traffic coming from untrusted desktops/devices and mark them accordingly. The service-policies generated do police. The following commands enable auto-QoS for trusted interfaces: • **auto qos trust cos**: Class of service. • **auto qos trust dscp**: Differentiated Services Code Point.
Step 4	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 5	**show auto qos interface** *interface-id* **Example:**	(Optional) Displays the auto-QoS command on the interface on which auto-QoS was enabled. Use the **show running-config** command to

	Command or Action	Purpose
	`Device# show auto qos interface HundredGigE 1/0/1`	display the auto-QoS configuration and user modifications.

Upgrading Auto-QoS

Before you begin

Prior to upgrading, you need to remove all auto-QoS configurations currently on the switch. This sample procedure describes that process.

After following this sample procedure, you must then reboot the switch with the new or upgraded software image and reconfigure auto-QoS.

Procedure

Step 1 **show auto qos**

Example:

```
Device# show auto qos

TwentyFiveGigE1/0/1
auto qos trust dscp

TwentyFiveGigE1/0/2
auto qos trust cos
```

Example:

On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

```
Device# show auto qos

TwentyFiveGigE1/0/1
auto qos trust dscp

TwentyFiveGigE1/0/2
auto qos trust cos
```

In privileged EXEC mode, record all current auto QoS configurations by entering this command.

Step 2 **no auto qos**

Example:

```
Device(config-if)#no auto qos
```

In interface configuration mode, run the appropriate **no auto qos** command on each interface that has an auto QoS configuration.

Step 3 **show running-config | i autoQos**

Example:

```
Device# show running-config | i autoQos
```

Return to privileged EXEC mode, and record any remaining auto QoS maps class maps, policy maps, access lists, table maps, or other configurations by entering this command.

Step 4 **no policy-map** *policy-map_name*

Example:

```
Device(config)# no policy-map pmap_101
Device(config)# no class-map cmap_101
Device(config)# no ip access-list extended AutoQos-101
Device(config)# no table-map 101
Device(config)# no table-map policed-dscp
```

In global configuration mode, remove the QoS class maps, policy maps, access-lists, table maps, and any other auto QoS configurations by entering these commands:

- **no policy-map** *policy-map-name*
- **no class-map** *class-map-name*
- **no ip access-list extended** *Auto-QoS-x*
- **no table-map** *table-map-name*
- **no table-map policed-dscp**

Step 5 **show running-config | i autoQos**

Example:

```
Device#show running-config | i autoQos
```

Return to privileged EXEC mode, run this command again to ensure that no auto-QoS configuration or remaining parts of the auto-QoS configuration exists

Step 6 **show auto qos**

Example:

```
Device# show auto qos
```

Run this command to ensure that no auto-QoS configuration or remaining parts of the configuration exists.

Step 7 **write memory**

Example:

```
Device# write memory
```

Write the changes to the auto QoS configuration to NV memory by entering the **write memory** command.

What to do next

Reboot the switch with the new or upgraded software image.

After rebooting with the new or upgraded software image, re-configure auto-QoS for the appropriate switch interfaces as determined by running the **show auto qos** command described in step 1.

Note There is only one table-map for exceed and another table-map for violate markdown per switch or stack. If the switch already has a table-map under the exceed action, then the auto-qos policy cannot be applied.

Enabling Auto-Qos Compact

To enable auto-QoS compact, enter this command:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**auto qos global compact** **Example:** `Device(config)# auto qos global compact`	Enables auto-Qos compact and generates (hidden) the global configurations for auto-QoS. You can then enter the auto-QoS command you want to configure in the interface configuration mode and the interface commands that the system generates are also hidden. To display the auto-QoS configuration that has been applied, use these the privileged EXEC commands: • **show derived-config** • **show policy-map** • **show access-list** • **show class-map** • **show table-map** • **show auto qos** • **show policy-map interface** • **show ip access-lists**

	Command or Action	Purpose
		• **show policy-map type queue** • **show policy-map type queueing interface** **Note** The **show policy-map type queueing interface** command is only applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2). These commands will have keyword "**AutoQos-**".

What to do next

To disable auto-QoS compact, remove auto-Qos instances from all interfaces by entering the **no** form of the corresponding auto-QoS commands and then enter the **no auto qos global compact** global configuration command.

Monitoring Auto-QoS

Table 1: Commands for Monitoring Auto-QoS

Command	Description
show auto qos [**interface** [*interface-id*]]	Displays the initial auto-QoS configuration. You can compare the **show auto qos** and the **show running-config** command output to identify the user-defined QoS settings.
show running-config	Displays information about the QoS configuration that might be affected by auto-QoS. You can compare the **show auto qos** and the **show running-config** command output to identify the user-defined QoS settings.
show derived-config	Displays the hidden **mls qos** command which get configured along with the running configs because of auto-qos template.

Troubleshooting Auto-QoS

To troubleshoot auto-QoS, use the **debug auto qos** privileged EXEC command. For more information, see the **debug auto qos** command in the command reference for this release.

To disable auto-QoS on a port, use the **no** form of the **auto qos** command interface configuration command, such as **no auto qos voip**. Only the auto-QoS-generated interface configuration commands for this port are removed. If this is the last port on which auto-QoS is enabled and you enter the **no auto qos voip** command, auto-QoS is considered disabled even though the auto-QoS-generated global configuration commands remain (to avoid disrupting traffic on other ports affected by the global configuration).

Configuration Examples for Auto-QoS

Example: auto qos trust cos

The following is an example of the **auto qos trust cos** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Cos-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

Additionally, on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2), the following class maps are created and applied when running this command:

- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Signaling-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Transaction-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Stream-Queue (match-any)
- class-default (match-any)

The following sample output does not apply to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2).

```
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos trust cos
Device(config-if)# end
Device# show policy-map interface HundredGigE1/0/2

HundredGigE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy

    Class-map: class-default (match-any)
      0 packets
      Match: any
      QoS Set
         cos cos table AutoQos-4.0-Trust-Cos-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      0 packets
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: cos  5
      Priority: 30% (7500000 kbps), burst bytes 187500000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      0 packets
      Match:  dscp cs2 (16) cs3 (24) cs6 (48) cs7 (56)
      Match: cos  3
      Queueing

      queue-limit dscp 16 percent 80
      queue-limit dscp 24 percent 90
      queue-limit dscp 48 percent 100
      queue-limit dscp 56 percent 100
      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
      0 packets
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Match: cos  4
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
```

```
    0 packets
    Match:  dscp af21 (18) af22 (20) af23 (22)
    Match: cos  2
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
    0 packets
    Match:  dscp af11 (10) af12 (12) af13 (14)
    Match: cos  1
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 4%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
    0 packets
    Match:  dscp cs1 (8)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 1%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
    0 packets
    Match:  dscp af31 (26) af32 (28) af33 (30)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    0 packets
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2), the **show policy-map interface** will only display the ingress policy information.

```
Device(config)# interface HundredGigE1/0/9
Device(config-if)# auto qos trust cos
Device(config-if)# end
Device# show policy-map interface HundredGigE1/0/9

HundredGigE1/0/9

Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy
```

```
Class-map: class-default (match-any)
0 packets
Match: any
QoS Set
traffic-class cos table AutoQos-4.0-Trust-Cos-Table
```

On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2), to view the egress queue information (output policy), use the **show policy-map type queueing interface** command.

```
Device(config)# interface HundredGigE1/0/9
Device(config-if)# auto qos trust cos
Device(config-if)# end
Device# show policy-map type queueing interface HunredGigE1 1/0/9

HundredGigE1/0/9

Service-policy queueing output: AutoQos-4.0-Output-Policy

queue stats for all priority classes:
Queueing
priority level 1
queue limit 96000 bytes
(total drops) 0
(bytes output) 381284209

Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
3562992 packets
Match: traffic-class 7
Priority: Strict,

Priority Level: 1
shape (average) cir 12000000000, bc 120000000, be 120000000
target shape rate 12000000000

Class-map: AutoQos-4.0-Output-Signaling-Queue (match-any)
0 packets
Match: traffic-class 6
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
0 packets
Match: traffic-class 5
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9
```

```
Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Transaction-Data-Queue (match-any)
0 packets
Match: traffic-class 4
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
0 packets
Match: traffic-class 3
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 4

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
0 packets
Match: traffic-class 2
Queueing
queue limit 30000000 bytes
(total drops) 0
```

```
(bytes output) 0
bandwidth remaining ratio 1

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Multimedia-Stream-Queue (match-any)
0 packets
Match: traffic-class 1
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: class-default (match-any)
3547910 packets
Match: any
Queueing
queue limit 75000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 23

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1
```

Example: auto qos trust dscp

The following is an example of the **auto qos trust dscp** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

Additionally, on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2), the following class maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Priority-Class1 (match-any)
- AutoQos-4.0-Trust-Dscp-Priority-Class2 (match-any)
- AutoQos-4.0-Trust-Dscp-Signaling-Class1 (match-any)
- AutoQos-4.0-Trust-Dscp-Signaling-Class2 (match-any)
- AutoQos-4.0-Trust-Dscp-Multimedia-Conf-Class1 (match-any)
- AutoQos-4.0-Trust-Dscp-Multimedia-Conf-Class2 (match-any)
- AutoQos-4.0-Trust-Dscp-Transaction-Data-Class1 (match-any)
- AutoQos-4.0-Trust-Dscp-Transaction-Data-Class2 (match-any)
- AutoQos-4.0-Trust-Dscp-Bulk-Data-Class1 (match-any)
- AutoQos-4.0-Trust-Dscp-Bulk-Data-Class2 (match-any)
- AutoQos-4.0-Trust-Dscp-Scavenger-Class (match-any)
- AutoQos-4.0-Trust-Dscp-Multimedia-Stream-Class1 (match-any)
- AutoQos-4.0-Trust-Dscp-Multimedia-Stream-Class2 (match-any)

The following sample output does not apply to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2).

```
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos trust dscp
Device(config-if)# end
Device#show policy-map interface HundredGigE1/0/2
```

```
HundredGigE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

    Class-map: class-default (match-any)
      0 packets
      Match: any
      QoS Set
        dscp dscp table AutoQos-4.0-Trust-Dscp-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      0 packets
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: dscp  5
      Priority: 30% (30000000 kbps), burst bytes 750000000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      0 packets
      Match:  dscp cs2 (16) cs3 (24) cs6 (48) cs7 (56)
      Match: dscp  3
      Queueing
      queue-limit dscp 16 percent 80
      queue-limit dscp 24 percent 90
      queue-limit dscp 48 percent 100
      queue-limit dscp 56 percent 100
      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
      0 packets
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Match: dscp  4
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
      0 packets
      Match:  dscp af21 (18) af22 (20) af23 (22)
      Match: dscp  2
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10
```

```
Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  0 packets
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: dscp  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  0 packets
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  0 packets
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  0 packets
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

On Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2)the **show policy-map interface** will only display the ingress policy information

```
Device(config)# interface HundredGigE1/0/9
Device(config-if)# auto qos trust dscp
Device(config-if)# end

Device# show policy-map interface HundredGigE1/0/9
Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

Class-map: AutoQos-4.0-Trust-Dscp-Priority-Class1 (match-any)
0 packets
Match: dscp ef (46)
QoS Set
traffic-class 7

Class-map: AutoQos-4.0-Trust-Dscp-Priority-Class2 (match-any)
0 packets
Match: dscp cs4 (32) cs5 (40)
QoS Set
```

```
traffic-class 7
discard-class 1

Class-map: AutoQos-4.0-Trust-Dscp-Signaling-Class1 (match-any)
0 packets
Match: dscp cs7 (56)
QoS Set
traffic-class 6

Class-map: AutoQos-4.0-Trust-Dscp-Signaling-Class2 (match-any)
0 packets
Match: dscp cs2 (16) cs3 (24) cs6 (48)
QoS Set
traffic-class 6
discard-class 1

Class-map: AutoQos-4.0-Trust-Dscp-Multimedia-Conf-Class1 (match-any)
0 packets
Match: dscp af41 (34)
QoS Set
traffic-class 5

Class-map: AutoQos-4.0-Trust-Dscp-Multimedia-Conf-Class2 (match-any)
0 packets
Match: dscp af42 (36) af43 (38)
QoS Set
traffic-class 5
discard-class 1

Class-map: AutoQos-4.0-Trust-Dscp-Transaction-Data-Class1 (match-any)
0 packets
Match: dscp af21 (18)
QoS Set
traffic-class 4

Class-map: AutoQos-4.0-Trust-Dscp-Transaction-Data-Class2 (match-any)
0 packets
Match: dscp af22 (20) af23 (22)
QoS Set
traffic-class 4
discard-class 1

Class-map: AutoQos-4.0-Trust-Dscp-Bulk-Data-Class1 (match-any)
0 packets
Match: dscp af11 (10)
QoS Set
traffic-class 3

Class-map: AutoQos-4.0-Trust-Dscp-Bulk-Data-Class2 (match-any)
0 packets
Match: dscp af12 (12) af13 (14)
QoS Set
traffic-class 3
discard-class 1

Class-map: AutoQos-4.0-Trust-Dscp-Scavenger-Class (match-any)
0 packets
Match: dscp cs1 (8)
QoS Set
traffic-class 2

Class-map: AutoQos-4.0-Trust-Dscp-Multimedia-Stream-Class1 (match-any)
0 packets
Match: dscp af31 (26)
```

```
QoS Set
traffic-class 1

Class-map: AutoQos-4.0-Trust-Dscp-Multimedia-Stream-Class2 (match-any)
0 packets
Match: dscp af32 (28) af33 (30)
QoS Set
traffic-class 1
discard-class 1

Class-map: class-default (match-any)
0 packets
Match: any
```

To view egress queue information (output policy), use the **show policy-map type queueing interface** command:

```
Device(config)# interface HundredGigE1/0/9
Device(config-if)# auto qos trust dscp
Device(config-if)# end
Device# show policy-map type queueing interface HunredGigE1 1/0/9


Service-policy queueing output: AutoQos-4.0-Output-Policy

queue stats for all priority classes:
Queueing
priority level 1
queue limit 96000 bytes
(total drops) 0
(bytes output) 6957

Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
49 packets
Match: traffic-class 7
Priority: Strict,

Priority Level: 1
shape (average) cir 30000000000, bc 300000000, be 300000000
target shape rate 30000000000

Class-map: AutoQos-4.0-Output-Signaling-Queue (match-any)
0 packets
Match: traffic-class 6
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
```

```
0 packets
Match: traffic-class 5
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Transaction-Data-Queue (match-any)
0 packets
Match: traffic-class 4
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
0 packets
Match: traffic-class 3
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 4

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1
```

```
Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
0 packets
Match: traffic-class 2
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 1

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Multimedia-Stream-Queue (match-any)
0 packets
Match: traffic-class 1
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: class-default (match-any)
50 packets
Match: any
Queueing
queue limit 75000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 23

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
```

```
6 0 0 1/1
7 0 0 1/1
```

Example: auto qos video cts

Note This example section is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

The following is an example of the **auto qos video cts** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Cos-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitEthernet1/0/2
Device(config-if)# auto qos video cts
Device(config-if)# end
Device# show policy-map interface HundredGigabitEthernet1/0/2

HundredGigabitEthernet1/0/2

  Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        cos cos table AutoQos-4.0-Trust-Cos-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0
```

```
Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
  Match:  dscp cs4 (32) cs5 (40) ef (46)
  Match: cos  5
  Priority: 30% (300000 kbps), burst bytes 7500000,

  Priority Level: 1

Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
  Match:  dscp cs3 (24) cs6 (48) cs7 (56)
  Match: cos  3
  Queueing
  queue-limit dscp 16 percent 80
  queue-limit dscp 24 percent 90
  queue-limit dscp 48 percent 100

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%

  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
  Match:  dscp af41 (34) af42 (36) af43 (38)
  Match: cos  4
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
  Match:  dscp af21 (18) af22 (20) af23 (22)
  Match: cos  2
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing
```

```
    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

Example: auto qos video ip-camera

Note This example section is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

The following is an example of the **auto qos video ip-camera** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# auto qos video ip-camera
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/2

HundredGigabitE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

    Class-map: class-default (match-any)
      Match: any
```

```
    QoS Set
      dscp dscp table AutoQos-4.0-Trust-Dscp-Table

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
    Queueing
    priority level 1

    (total drops) 0
    (bytes output) 0

  Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
    Match:  dscp cs4 (32) cs5 (40) ef (46)
    Match: cos  5
    Priority: 30% (300000 kbps), burst bytes 7500000,

    Priority Level: 1

  Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
    Match:  dscp cs3 (24) cs6 (48) cs7 (56)
    Match: cos  3
    Queueing
    queue-limit dscp 16 percent 80
    queue-limit dscp 24 percent 90
    queue-limit dscp 48 percent 100

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%

    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
    Match:  dscp af41 (34) af42 (36) af43 (38)
    Match: cos  4
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
    Match:  dscp af21 (18) af22 (20) af23 (22)
    Match: cos  2
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
    Match:  dscp af11 (10) af12 (12) af13 (14)
    Match: cos  1
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 4%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
```

```
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

Example: auto qos video media-player

Note This example section is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

The following is an example of the **auto qos video media-player** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Dscp-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)

- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# auto qos video media-player
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/2

HundredGigabitE1/0/2

  Service-policy input: AutoQos-4.0-Trust-Dscp-Input-Policy

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        dscp dscp table AutoQos-4.0-Trust-Dscp-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: cos  5
      Priority: 30% (300000 kbps), burst bytes 7500000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      Match:  dscp cs3 (24) cs6 (48) cs7 (56)
      Match: cos  3
      Queueing
      queue-limit dscp 16 percent 80
      queue-limit dscp 24 percent 90
      queue-limit dscp 48 percent 100

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Match: cos  4
      Queueing

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%
      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
      Match:  dscp af21 (18) af22 (20) af23 (22)
      Match: cos  2
      Queueing

      (total drops) 0
      (bytes output) 0
```

```
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

Example: auto qos voip trust

The following is an example of the **auto qos voip trust** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-Trust-Cos-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)

- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

The following sample output is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2).

```
Device(config)# interface HundredGigabitE1/0/3
Device(config-if)# auto qos voip trust
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/3

HundredGigabitE1/0/3

  Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy

    Class-map: class-default (match-any)
      Match: any
      QoS Set
        cos cos table AutoQos-4.0-Trust-Cos-Table

  Service-policy output: AutoQos-4.0-Output-Policy

    queue stats for all priority classes:
      Queueing
      priority level 1

      (total drops) 0
      (bytes output) 0

    Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
      Match:  dscp cs4 (32) cs5 (40) ef (46)
      Match: cos  5
      Priority: 30% (300000 kbps), burst bytes 7500000,

      Priority Level: 1

    Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
      Match:  dscp cs3 (24) cs6 (48) cs7 (56)
      Match: cos  3
      Queueing
      queue-limit dscp 16 percent 80
      queue-limit dscp 24 percent 90
      queue-limit dscp 48 percent 100

      (total drops) 0
      (bytes output) 0
      bandwidth remaining 10%

      queue-buffers ratio 10

    Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
      Match:  dscp af41 (34) af42 (36) af43 (38)
      Match: cos  4
      Queueing

      (total drops) 0
      (bytes output) 0
```

```
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
    Match:  dscp af21 (18) af22 (20) af23 (22)
    Match: cos  2
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
    Match:  dscp af11 (10) af12 (12) af13 (14)
    Match: cos  1
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 4%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
    Match:  dscp cs1 (8)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 1%
    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
    Match:  dscp af31 (26) af32 (28) af33 (30)
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%
    queue-buffers ratio 10

  Class-map: class-default (match-any)
    Match: any
    Queueing

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 25%
    queue-buffers ratio 25
```

The following is a sample output of the **show policy-map interface** command for a Layer 2 interface from the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2). Depending on the type of interface (Layer 3 or Layer 2), QoS policies from trust DSCP or CoS will be installed.

```
Device(config)# interface HundredGigE1/0/9
Device(config-if)# auto qos voip trust
Device(config-if)# end
Device# show policy-map interface HundredGigE1 1/0/9

HundredGigE1/0/9

Service-policy input: AutoQos-4.0-Trust-Cos-Input-Policy
```

```
Class-map: class-default (match-any)
0 packets
Match: any
QoS Set
traffic-class cos table AutoQos-4.0-Trust-Cos-Table
```

The following is a sample output of the **show policy-map type queueing interface** command from the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2).

```
Device(config)# interface TwentyFiveGigE 6/0/1
Device(config-if)# auto qos voip trust
Device(config-if)# end
Device# show policy-map type queueing interface TwentyFiveGigE 6/0/1

TwentyFiveGigE6/0/1

Service-policy queueing output: AutoQos-4.0-Output-Policy

queue stats for all priority classes:
Queueing
priority level 1
queue limit 96000 bytes
(total drops) 0
(bytes output) 0

Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
0 packets
Match: traffic-class 7
Priority: Strict,

Priority Level: 1
shape (average) cir 15000000000, bc 150000000, be 150000000
target shape rate 15000000000

Class-map: AutoQos-4.0-Output-Signaling-Queue (match-any)
0 packets
Match: traffic-class 6
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
0 packets
Match: traffic-class 5
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
```

```
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Transaction-Data-Queue (match-any)
0 packets
Match: traffic-class 4
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
0 packets
Match: traffic-class 3
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 4

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
0 packets
Match: traffic-class 2
Queueing
queue limit 30000000 bytes
```

```
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 1

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: AutoQos-4.0-Output-Multimedia-Stream-Queue (match-any)
0 packets
Match: traffic-class 1
Queueing
queue limit 30000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 9

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1

Class-map: class-default (match-any)
0 packets
Match: any
Queueing
queue limit 75000000 bytes
(total drops) 0
(bytes output) 0
bandwidth remaining ratio 23

Exp-weight-constant: 1 (1/2)
Mean queue depth: 0
Minimum Maximum Mark
thresh thresh prob
0 85 100 1/1
1 65 100 1/1
2 0 0 1/1
3 0 0 1/1
4 0 0 1/1
5 0 0 1/1
6 0 0 1/1
7 0 0 1/1
```

Example: auto qos voip cisco-phone

Note This example section is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

The following is an example of the **auto qos voip cisco-phone** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-CiscoPhone-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- AutoQos-4.0-Voip-Data-Class (match-any)
- AutoQos-4.0-Voip-Signal-Class (match-any)
- AutoQos-4.0-Default-Class (match-any)
- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigabitE1/0/5
Device(config-if)# auto qos voip cisco-phone
Device(config-if)# end
Device# show policy-map interface HundredGigabitE1/0/5

 HundredGigabitE1/0/5

  Service-policy input: AutoQos-4.0-CiscoPhone-Input-Policy

    Class-map: AutoQos-4.0-Voip-Data-Class (match-any)
      Match: ip dscp ef (46)
      QoS Set
        ip dscp ef
      police:
          cir 128000 bps, bc 8000 bytes, be 8000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          set-dscp-transmit dscp table policed-dscp
        violated 0 bytes; actions:
          drop
```

```
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: AutoQos-4.0-Voip-Signal-Class (match-any)
    Match: ip dscp cs3 (24)
    QoS Set
      ip dscp cs3
    police:
        cir 32000 bps, bc 8000 bytes, be 8000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: AutoQos-4.0-Default-Class (match-any)
    Match: access-group name AutoQos-4.0-Acl-Default
    QoS Set
      dscp default
    police:
        cir 10000000 bps, bc 8000 bytes, be 8000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: class-default (match-any)
    Match: any

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
    Queueing
    priority level 1

    (total drops) 0
    (bytes output) 0

  Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
    Match:  dscp cs4 (32) cs5 (40) ef (46)
    Match: cos  5
    Priority: 30% (300000 kbps), burst bytes 7500000,

    Priority Level: 1

  Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
    Match:  dscp cs3 (24) cs6 (48) cs7 (56)
    Match: cos  3
    Queueing
    queue-limit dscp 16 percent 80
    queue-limit dscp 24 percent 90
    queue-limit dscp 48 percent 100

    (total drops) 0
    (bytes output) 0
    bandwidth remaining 10%

    queue-buffers ratio 10

  Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
```

```
  Match:  dscp af41 (34) af42 (36) af43 (38)
  Match: cos  4
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
  Match:  dscp af21 (18) af22 (20) af23 (22)
  Match: cos  2
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
  Match: any
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 25%
  queue-buffers ratio 25
```

Example: auto qos voip cisco-softphone

Note This example section is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

The following is an example of the **auto qos voip cisco-softphone** command and the applied policies and class maps.

The following policy maps are created and applied when running this command:

- AutoQos-4.0-CiscoSoftPhone-Input-Policy
- AutoQos-4.0-Output-Policy

The following class maps are created and applied when running this command:

- AutoQos-4.0-Voip-Data-Class (match-any)
- AutoQos-4.0-Voip-Signal-Class (match-any)
- AutoQos-4.0-Multimedia-Conf-Class (match-any)
- AutoQos-4.0-Bulk-Data-Class (match-any)
- AutoQos-4.0-Transaction-Class (match-any)
- AutoQos-4.0-Scavanger-Class (match-any)
- AutoQos-4.0-Signaling-Class (match-any)
- AutoQos-4.0-Default-Class (match-any)
- class-default (match-any)
- AutoQos-4.0-Output-Priority-Queue (match-any)
- AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
- AutoQos-4.0-Output-Trans-Data-Queue (match-any)
- AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
- AutoQos-4.0-Output-Scavenger-Queue (match-any)
- AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)

```
Device(config)# interface HundredGigE1/0/20
Device(config-if)# auto qos voip cisco-softphone
Device(config-if)# end
Device# show policy-map interface HundredGigE1/0/20

 HundredGigE1/0/20

  Service-policy input: AutoQos-4.0-CiscoSoftPhone-Input-Policy

    Class-map: AutoQos-4.0-Voip-Data-Class (match-any)
      Match: ip dscp ef (46)
```

```
  QoS Set
    ip dscp ef
  police:
      cir 128000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Voip-Signal-Class (match-any)
  Match: ip dscp cs3 (24)
  QoS Set
    ip dscp cs3
  police:
      cir 32000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Multimedia-Conf-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-MultiEnhanced-Conf
  QoS Set
    dscp af41
  police:
      cir 5000000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Bulk-Data-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Bulk-Data
  QoS Set
    dscp af11
  police:
      cir 10000000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
    violated 0 bytes; actions:
      drop
    conformed 0000 bps, exceed 0000 bps, violate 0000 bps

Class-map: AutoQos-4.0-Transaction-Class (match-any)
  Match: access-group name AutoQos-4.0-Acl-Transactional-Data
  QoS Set
    dscp af21
  police:
      cir 10000000 bps, bc 8000 bytes, be 8000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      set-dscp-transmit dscp table policed-dscp
```

```
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: AutoQos-4.0-Scavanger-Class (match-any)
    Match: access-group name AutoQos-4.0-Acl-Scavanger
    QoS Set
      dscp cs1
    police:
        cir 10000000 bps, bc 8000 bytes, be 8000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: AutoQos-4.0-Signaling-Class (match-any)
    Match: access-group name AutoQos-4.0-Acl-Signaling
    QoS Set
      dscp cs3
    police:
        cir 32000 bps, bc 8000 bytes, be 8000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: AutoQos-4.0-Default-Class (match-any)
    Match: access-group name AutoQos-4.0-Acl-Default
    QoS Set
      dscp default
    police:
        cir 10000000 bps, bc 8000 bytes, be 8000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        set-dscp-transmit dscp table policed-dscp
      violated 0 bytes; actions:
        drop
      conformed 0000 bps, exceed 0000 bps, violate 0000 bps

  Class-map: class-default (match-any)
    Match: any

Service-policy output: AutoQos-4.0-Output-Policy

  queue stats for all priority classes:
    Queueing
    priority level 1

    (total drops) 0
    (bytes output) 0

  Class-map: AutoQos-4.0-Output-Priority-Queue (match-any)
    Match:  dscp cs4 (32) cs5 (40) ef (46)
    Match: cos  5
    Priority: 30% (300000 kbps), burst bytes 7500000,

    Priority Level: 1
```

```
Class-map: AutoQos-4.0-Output-Control-Mgmt-Queue (match-any)
  Match:  dscp cs3 (24) cs6 (48) cs7 (56)
  Match: cos  3
  Queueing
  queue-limit dscp 16 percent 80
  queue-limit dscp 24 percent 90
  queue-limit dscp 48 percent 100

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%

  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Conf-Queue (match-any)
  Match:  dscp af41 (34) af42 (36) af43 (38)
  Match: cos  4
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Trans-Data-Queue (match-any)
  Match:  dscp af21 (18) af22 (20) af23 (22)
  Match: cos  2
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Bulk-Data-Queue (match-any)
  Match:  dscp af11 (10) af12 (12) af13 (14)
  Match: cos  1
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 4%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Scavenger-Queue (match-any)
  Match:  dscp cs1 (8)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 1%
  queue-buffers ratio 10

Class-map: AutoQos-4.0-Output-Multimedia-Strm-Queue (match-any)
  Match:  dscp af31 (26) af32 (28) af33 (30)
  Queueing

  (total drops) 0
  (bytes output) 0
  bandwidth remaining 10%
  queue-buffers ratio 10

Class-map: class-default (match-any)
```

```
Match: any
Queueing

(total drops) 0
(bytes output) 0
bandwidth remaining 25%
queue-buffers ratio 25
```

Example: auto qos global compact

The following is an example of the **auto qos global compact** command. This example is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP- 2).

```
Device# configure terminal
Device(config)# auto qos global compact
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos voip cisco-phone

Device# show auto qos
HundredGigE1/0/2
auto qos voip cisco-phone

Device# show running-config interface HundredGigE1/0/2
interface HundredGigE1/0/2
auto qos voip cisco-phone
end
```

The following is an example of the **auto qos global compact** command from the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

```
Device# configure terminal
Device(config)# auto qos global compact
Device(config)# interface HundredGigE1/0/2
Device(config-if)# auto qos voip trust

Device# show auto qos
HundredGigE1/0/2
auto qos voip trust

Device# show running-config interface HundredGigE1/0/2
interface HundredGigE1/0/2
auto qos voip trust
end
```

Where to Go Next for Auto-QoS

Review the QoS documentation if you require any specific QoS changes to your auto-QoS configuration.

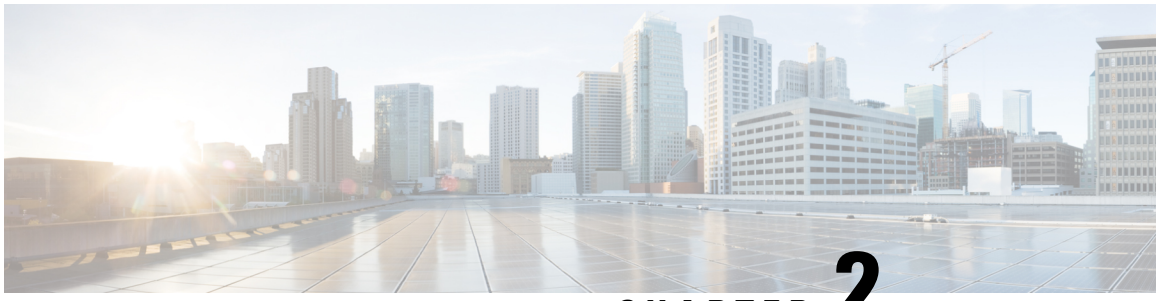
Feature History for Auto-QoS

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Auto-QoS	Auto-QoS feature simplifies the deployment of QoS features. This feature determines the network design and enables QoS configurations so that the switch can prioritize different traffic flows.
Cisco IOS XE Cupertino 17.7.1	Auto-QoS	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to Cisco Feature Navigator.



CHAPTER 2

Configuring QoS

Prerequisites for QoS

Before configuring standard Quality of Service (QoS), you must have a thorough understanding of these items:

- Standard QoS concepts.
- Classic Cisco IOS QoS.
- Modular QoS CLI (MQC).
- Understanding of QoS implementation.
- The types of applications used and the traffic patterns on your network.
- Traffic characteristics and needs of your network. For example, is the traffic on your network bursty? Do you need to reserve bandwidth for voice and video streams?
- Bandwidth requirements and speed of the network.
- Location of congestion points in the network.

Restrictions for QoS on Wired Targets

A target is an entity where a policy is applied. A wired target can be either a port or VLAN.

The following are restrictions for applying QoS features on the device for the wired target:

- A maximum of 8 queuing classes are supported on the device port for the wired target.
- A maximum of 63 policers are supported per policy on the wired port for the wired target, in the ingress or egress directions.
- A maximum of 1599 policy-maps can be created.
- Starting with Cisco IOS XE Amsterdam 17.3.6, all fragmented traffic is assigned to the default class **(class class-default)**. This restriction does not apply to the Cisco Catalyst 9600X Series switches.
- No more than two levels are supported in a QoS hierarchy.
- In a hierarchical policy, overlapping actions between parent and child are not allowed, except when a policy has the port shaper in the parent and queuing features in the child policy.
- A QoS policy cannot be attached to any EtherChannel interface.
- Policing in both the parent and child is not supported in a QoS hierarchy.
- Marking in both the parent and child is not supported in a QoS hierarchy.
- With shaping, there is an IPG overhead of 20Bytes for every packet that is accounted internally in the hardware. Shaping accuracy will be effected by this, specially for packets of small size.
- Empty classes are supported.
- A maximum of 256 classes are supported per policy on the wired port for the wired target.
- Based on the Cisco UADP architecture, traffic is subjected to QoS lookup and the corresponding configured actions even if this traffic is later dropped in the Egress Global Resolution block and is never transmitted out of the actual interface.
- The actions under a policer within a policy map have the following restrictions:
    - The conform action must be transmit.
- Only marking policy is supported on SVI.
- A port-level input marking policy takes precedence over an SVI policy; however, if no port policy is configured, the SVI policy takes precedence. For a port policy to take precedence, define a port-level policy; so that the SVI policy is overwritten.
- Classification counters have the following specific restrictions:
    - Classification counters count packets instead of bytes.
    - Filter-based classification counters are not supported
    - Only QoS configurations with marking or policing trigger the classification counter.
    - The classification counter is not port based. This means that the classification counter aggregates all packets belonging to the same class of the same policy which attach to different interfaces.
    - As long as there is policing or marking action in the policy, the class will have classification counters.
    - Classification counters are not supported on pure queuing policies under any class-map.
    - When there are multiple match statements in a class, the traffic counter is cumulative for all the match statements in the class.

- Classification counters (class-map) are not available in queuing policy with actions like bandwidth, WRED, queue-buffer, shaping, and so on. The **show policy-map interface** command output will display classification counters (class-map) only for policies having either remarking or policer action.

- Egress remarking or set option is supported based on DSCP, CoS, precedence, MPLS EXP, or QoS-group. A new policy-map must be defined and applied on the interface in egress direction. In case of MPLS, the egress remarking policy on label imposition node works only if there is an EXP based marking policy on the ingress interface.
- The device supports a total of eight table maps for policer exceed markdown and eight table maps for policer violate markdown.
- Hierarchical policies are required for the following:
  - Port-shapers
  - Aggregate policers
  - PV policy
  - Parent shaping and child marking/policing
- In a HQoS policy with parent shaping and child policy having priority level queuing and priority level policing, the statistics for policing are not updated. Only QoS shaper statistics are updated. To view the QoS shaper statistics, use the **show policy-map interface** command in global configuration mode.
- For ports with wired targets, these are the only supported hierarchical policies:
  - Police chaining in the same policy is unsupported.
  - Hierarchical queuing is unsupported in the same policy (port shaper is the exception).
  - In a parent class, all filters must have the same type. The child filter type must match the parent filter type with the following exceptions:
    - If the parent class is configured to match IP, then the child class can be configured to match the ACL.
    - If the parent class is configured to match CoS, then the child class can be configured to match the ACL.
- The **trust device** *device_type* command available in interface configuration mode is a stand-alone command on the device. When using this command in an AutoQoS configuration, if the connected peer device is not a corresponding device (defined as a device matching your trust policy), both CoS and DSCP values are set to "0" and any input policy will not take effect. If the connected peer device is a corresponding device, input policy will take effect.
- Queuing actions are supported only on DSCP, CoS, QoS-group, IP precedence, and EXP based classification.

The following are restrictions for applying QoS features on the VLAN to the wired target:

- For a flat or nonhierarchical policy, only marking or a table map is supported.

The following are restrictions and considerations for applying QoS features on EtherChannel and channel member interfaces:

- QoS is not supported on an EtherChannel interface.
- QoS is supported on EtherChannel member interfaces in both ingress and egression directions. All EtherChannel members must have the same QoS policy applied. If the QoS policy is not the same, each individual policy on the different link acts independently.
- On attaching a service policy to channel members, the following warning message appears to remind the user to make sure the same policy is attached to all ports in the EtherChannel: ' Warning: add service policy will cause inconsistency with port xxx in ether channel xxx. '.
- Auto QoS is not supported on EtherChannel members.

Note On attaching a service policy to an EtherChannel, the following message appears on the console: ' Warning: add service policy will cause inconsistency with port xxx in ether channel xxx. '. This warning message should be expected. This warning message is a reminder to attach the same policy to other ports in the same EtherChannel. The same message will be seen during boot up. This message does not mean there is a discrepancy between the EtherChannel member ports.

Information About QoS

The following sections provide information about QoS.

QoS Components

Quality of service (QoS) consists of the following key components:

- Classification: Classification is the process of distinguishing one type of traffic from another based upon access control lists (ACLs), Differentiated Services Code Point (DSCP), Class of Service (CoS), and other factors.
- Marking and mutation: Marking is used on traffic to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another. When traffic is marked, QoS operations on that traffic can be applied. This can be accomplished directly using the **set** command or through a table map, which takes input values and translates them directly to values on output.
- Shaping and policing: Shaping is the process of imposing a maximum rate of traffic, while regulating the traffic rate in such a way that downstream devices are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface. Policing is used to impose a maximum rate on a traffic class. If the rate is exceeded, then a specific action is taken as soon as the event occurs.
- Queuing: Queuing is used to prevent traffic congestion. Traffic is sent to specific queues for servicing and scheduling based upon bandwidth allocation. Traffic is then scheduled or sent out through the port.
- Bandwidth: Bandwidth allocation determines the available capacity for traffic that is subject to QoS policies.

- Trust: Trust enables traffic to pass through the device, and the Differentiated Services Code Point (DSCP), precedence, or CoS values coming in from the end points are retained in the absence of any explicit policy configuration.

QoS Terminology

The following terms are used interchangeably in this QoS configuration guide:

- Upstream (direction towards the device) is the same as ingress.
- Downstream (direction from the device) is the same as egress.

Information About QoS

By configuring the quality of service (QoS), you can provide preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size. The device sends the packets without any assurance of reliability, delay bounds, or throughput.

The following are specific features provided by QoS:

- Low latency
- Bandwidth guarantee
- Buffering capabilities and dropping disciplines
- Traffic policing
- Enables the changing of the attribute of the frame or packet header
- Relative services

Modular QoS CLI

QoS features are enabled through the Modular QoS CLI (MQC). The MQC is a CLI structure that allows you to create traffic policies and attach these policies to interfaces. A traffic policy contains a traffic class and one or more QoS features. A traffic class is used to classify traffic, while the QoS features in the traffic policy determine how to treat the classified traffic. One of the main goals of MQC is to provide a platform-independent interface for configuring QoS across Cisco platforms.

QoS Wired Access Features

The following table describes the supported QoS features for wired access.

Table 2: QoS Wired Access Features

Feature	Description
Supported targets	• Gigabit Ethernet • 10-Gigabit Ethernet • 40-Gigabit Ethernet • 25-Gigabit Ethernet • 100-Gigabit Ethernet • EtherChannel member ports • SVI
Configuration sequence	QoS policy installed using the **service-policy** command.
Supported number of queues at port level	Up to eight queues supported on a port.

Supported classification mechanism	• DSCP
	• IP precedence
	• CoS
	• QoS-group
	• ACL membership including:
	  • IPv4 ACLs
	  • IPv6 ACLS
	  • MAC ACLs

Hierarchical QoS

The device supports hierarchical QoS (HQoS). HQoS allows you to perform:

- Hierarchical classification: Traffic classification is based upon other classes.
- Hierarchical policing: The process of having the policing configuration at multiple levels in a hierarchical policy.
- Hierarchical shaping: Shaping can also be configured at multiple levels in the hierarchy.

Note Hierarchical shaping is only supported for the port shaper, where for the parent you only have a configuration for the class default, and the only action for the class default is shaping.

QoS Implementation

Typically, networks operate on a best-effort delivery basis, which means that all traffic has equal priority and an equal chance of being delivered in a timely manner. When congestion occurs, all traffic has an equal chance of being dropped.

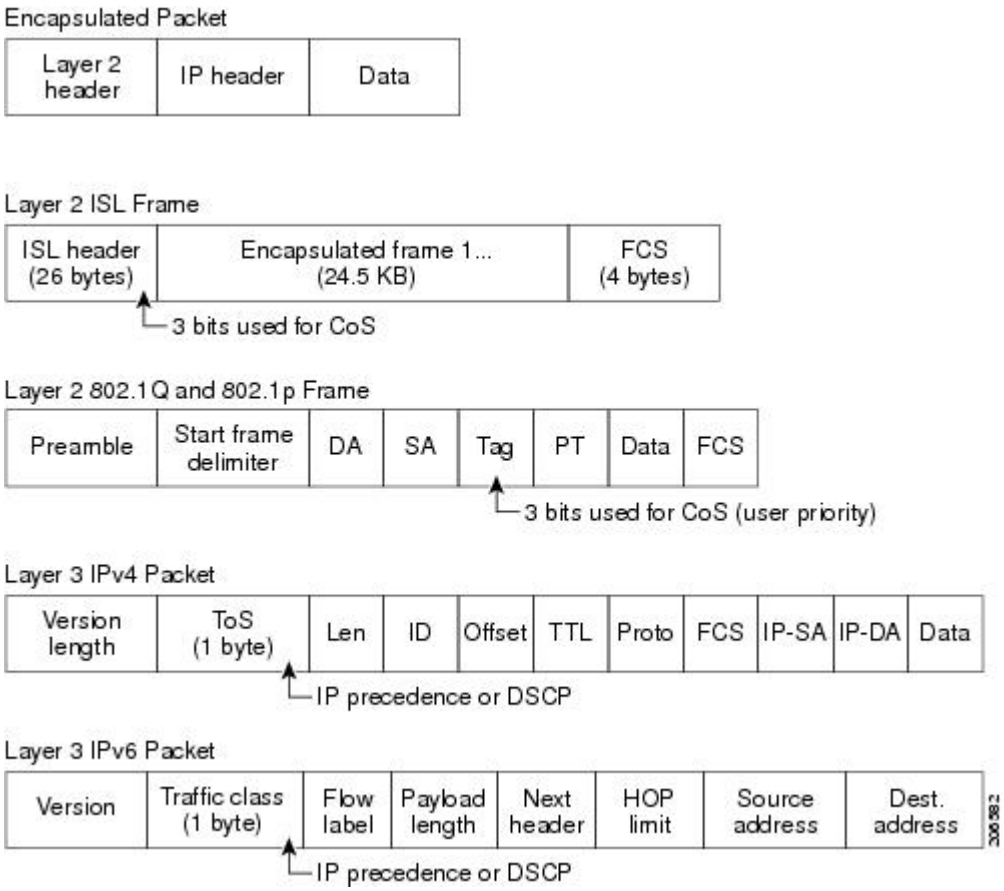
When you configure the QoS feature, you can select specific network traffic, prioritize it according to its relative importance, and use congestion-management and congestion-avoidance techniques to provide preferential treatment. Implementing QoS in your network makes network performance more predictable and bandwidth utilization more effective.

QoS implementation is based on the Differentiated Services (Diff-Serv) architecture, a standard from the Internet Engineering Task Force (IETF). This architecture specifies that each packet is classified upon entry into the network.

The classification is carried in the IP packet header, using six bits from the deprecated IP type of service (ToS) field to carry the classification (*class*) information. Classification can also be carried in the Layer 2 frame.

The special bits in the Layer 2 frame or a Layer 3 packet are shown in the following figure:

Figure 1: Classification Layers of QoS in Frames and Packets



Layer 2 Frame Prioritization Bits

Layer 2 Inter-Switch Link (ISL) frame headers have a 1-byte User field that carries an IEEE 802.1p class of service (CoS) value in the three least-significant bits. On ports configured as Layer 2 ISL trunks, all the traffic is in ISL frames.

Layer 2 802.1Q frame headers have a 2-byte Tag Control Information field that carries the CoS value in the three most-significant bits, which are called User Priority bits. On ports configured as Layer 2 802.1Q trunks, all the traffic is in 802.1Q frames, except for the traffic in the native VLAN.

Other frame types cannot carry Layer 2 CoS values.

Layer 2 CoS values range from 0 for low priority to 7 for high priority.

Layer 3 Packet Prioritization Bits

Layer 3 IP packets can carry either an IP precedence value or a Differentiated Services Code Point (DSCP) value. QoS supports the use of either value because DSCP values are backward-compatible with IP precedence values.

IP precedence values range from 0 to 7. DSCP values range from 0 to 63.

End-to-End QoS Solution Using Classification

All the switches and the routers that access the internet rely on class information to provide the same forwarding treatment to packets with the same class information, and different treatment to packets with different class information. The class information in the packet can be assigned by end hosts or by switches or routers along the way, based on a configured policy, detailed examination of the packet, or both. Detailed examination of the packet is expected to occur closer to the edge of the network, so that the core switches and routers are not overloaded with this task.

Switches and routers along the path can use the class information to limit the amount of resources allocated per traffic class. The behavior of an individual device when handling traffic in the Diff-Serv architecture is called per-hop behavior. If all the devices along a path provide a consistent per-hop behavior, you can construct an end-to-end QoS solution.

Implementing QoS in your network can be a simple task or a complex task, depending on the QoS features offered by your internetworking devices, the traffic types and patterns in your network, and the granularity of control that you need over incoming and outgoing traffic.

Packet Classification

Packet classification is the process of identifying a packet as belonging to one of several classes in a defined policy, based on certain criteria. The Modular QoS CLI (MQC) is a policy-class based language. The policy class language is used to define the following:

- Class map template with one or several match criteria
- Policy map template with one or several classes associated to the policy map

The policy map template is then associated to one or several interfaces on the device.

Packet classification is the process of identifying a packet as belonging to one of the classes defined in the policy map. The process of classification will exit when the packet being processed matches a specific filter in a class. This is referred to as first-match exit. If a packet matches multiple classes in a policy, irrespective of the order of classes in the policy map, it will still exit the classification process after matching the first class.

If a packet does not match any of the classes in the policy, it is classified into the default class in the policy. Every policy map has a default class, which is a system-defined class to match the packets that do not match any of the user-defined classes.

Packet classification can be categorized into the following types:

- Classification based on information that is propagated with the packet
- Classification based on information that is device specific
- Hierarchical classification

Classification Based on Information Propagated with a Packet

Classification based on information that is part of a packet, and propagated either end-to-end or between hops, typically includes the following:

- Classification based on Layer 3 or 4 headers
- Classification based on Layer 2 information

Classification Based on Layer 3 or Layer 4 Header

This is the most common deployment scenario. Numerous fields in the Layer 3 and Layer 4 headers can be used for packet classification.

At the most granular level, this classification methodology can be used to match an entire flow. For this deployment type, an access control list (ACL) can be used. ACLs can also be used based on various subsets of the flow, for example, source IP address only, destination IP address only, or a combination of both.

Classification can also be done based on the precedence or DSCP values in the IP header. The IP precedence field is used to indicate the relative priority with which a particular packet needs to be handled. It is made up of three bits in the IP header's type of service (ToS) byte.

The following table shows the different IP precedence bit values and their names.

Table 3: IP Precedence Values and Names

IP Precedence Value	IP Precedence Bits	IP Precedence Names
0	000	Routine
1	001	Priority
2	010	Immediate
3	011	Flash
4	100	Flash Override
5	101	Critical
6	110	Internetwork control
7	111	Network control

Note All the routing control traffic in a network use the IP precedence value 6 by default. IP precedence value 7 is also reserved for network control traffic. Therefore, we do not recommend the use of IP precedence values 6 and 7 for user traffic.

The DSCP field is made up of 6 bits in the IP header, and is being standardized by the Internet Engineering Task Force (IETF) Differentiated Services Working Group. The original ToS byte, which contained the DSCP bits, has been renamed the DSCP byte. The DSCP field is part of the IP header, similar to IP precedence. The DSCP field is a super set of the IP precedence field. Therefore, the DSCP field is used and is set in ways that are similar to what was described with respect to IP precedence.

Note

- The DSCP field definition is backward-compatible with the IP precedence values.
- Some fields in the Layer 2 header can also be set using a policy.

Classification Based on Layer 2 Header

A variety of methods can be used to perform classification based on the Layer 2 header information. The most common methods include the following:

- MAC address-based classification (only for access groups): Classification is based upon the source MAC address (for policies in the input direction) and destination MAC address (for policies in the output direction).
- Class-of-Service: Classification is based on the 3 bits in the Layer 2 header based on the IEEE 802.1p standard. This usually maps to the ToS byte in the IP header.
- VLAN ID: Classification is based on the VLAN ID of the packet.

Note Some of these fields in the Layer 2 header can also be set using a policy.

Classification Based on Information that is Device Specific

A device also provides classification mechanisms in scenarios that are available where classification is not based on information in the packet header or payload.

At times you might be required to aggregate traffic coming from multiple input interfaces into a specific class in the output interface. For example, multiple customer edge routers might be going into the same access device on different interfaces. The service provider might want to police all the aggregate voice traffic going into the core to a specific rate. However, the voice traffic coming in from the different customers could have different ToS settings. QoS group-based classification is a feature that is useful in these scenarios.

Policies configured on the input interfaces set the QoS group to a specific value, which can then be used to classify the packets in the policies enabled on the output interface.

The QoS group is a field in the packet data structure that is internal to a device. It is important to note that a QoS group is an internal label to the device and is not a part of the packet header.

QoS Wired Model

To implement QoS, the device must perform the following tasks:

- Traffic classification: Distinguish packets or flows from one another.
- Traffic marking and policing: Assign a label to indicate the given quality of service as the packets move through the device, and then make the packets comply with the configured resource usage limits.
- Queuing and scheduling: Provide different treatment in all the situations where resource contention exists.
- Shaping: Ensure that the traffic sent from the device meets a specific traffic profile.

Ingress Port Activity

The following activities occur at the ingress port of a device:

- Classification: Classifying a distinct path for a packet by associating it with a QoS label. For example, the device maps the CoS or DSCP in the packet to a QoS label to distinguish one type of traffic from another. The QoS label that is generated identifies all future QoS actions to be performed on this packet.

- Policing: Policing determines whether a packet is in or out of profile by comparing the rate of the incoming traffic to the configured policer. The policer limits the bandwidth consumed by a flow of traffic. The result is passed to the marker.
- Marking: Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and determines what to do with the packet (pass through a packet without modification, mark down the QoS label in the packet, or drop the packet).

Egress Port Activity

The following activities occur at the egress port of the device:

- Policing—Policing determines whether a packet is in or out of profile by comparing the rate of the incoming traffic to the configured policer. The policer limits the bandwidth consumed by a flow of traffic. The result is passed to the marker.
- Marking—Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and determines what to do with the packet (pass through a packet without modification, mark down the QoS label in the packet, or drop the packet).
- Queuing—Queuing evaluates the QoS packet label and the corresponding DSCP or CoS value before selecting which of the egress queues to use. Because congestion can occur when multiple ingress ports simultaneously send data to an egress port, Weighted Tail Drop (WTD) differentiates traffic classes and subjects the packets to different thresholds based on the QoS label. If the threshold is exceeded, the packet is dropped.

Classification

Classification is the process of distinguishing one kind of traffic from another by examining the fields in the packet. Classification is enabled only if QoS is enabled on a device. By default, QoS is enabled in a device.

During classification, a device performs a lookup and assigns a QoS label to a packet. The QoS label identifies all the QoS actions to be performed on a packet and from which queue the packet is sent.

Access Control Lists

You can use IP standard, IP extended, or Layer 2 MAC ACLs to define a group of packets with the same characteristics (class). You can also classify IP traffic based on IPv6 ACLs.

In the QoS context, the permit and deny actions in the access control entries (ACEs) have different meanings from security ACLs:

- If a match with a permit action is encountered (first-match principle), the specified QoS-related action is taken.
- If a match with a deny action is encountered, the ACL being processed is skipped, and the next ACL is processed.
- If no match with a permit action is encountered and all the ACEs have been examined, no QoS processing occurs on the packet, and the device offers best-effort service to the packet.
- If multiple ACLs are configured on a port, the lookup stops after the packet matches the first ACL with a permit action, and QoS processing begins.

Note When creating an access list, note that by default the end of the access list contains an implicit deny statement for everything if it did not find a match before reaching the end.

After a traffic class has been defined with the ACL, you can attach a policy to it. A policy might contain multiple classes with actions specified for each one of them. A policy might include commands to classify the class as a particular aggregate (for example, assign a DSCP) or rate-limit the class. This policy is then attached to a particular port on which it becomes effective.

You implement IP ACLs to classify IP traffic by using the **access-list** global configuration command; you implement Layer 2 MAC ACLs to classify non-IP traffic by using the **mac access-list extended** global configuration command.

Class Maps

A class map is a mechanism that you use to name a specific traffic flow (or class) and isolate it from all other traffic. The class map defines the criteria used to match against a specific traffic flow to further classify it. The criteria can include matching the access group defined by the ACL or matching a specific list of DSCP or IP precedence values or CoS values. If you have more than one type of traffic that you want to classify, you can create another class map and use a different name. After a packet is matched against the class-map criteria, you further classify it through the use of a policy map.

Note You cannot configure IPv4 and IPv6 classification criteria simultaneously in the same class-map. However, they can be configured in different class-maps in the same policy.

You create a class map by using the **class-map** global configuration command or the **class** policy-map configuration command. You should use the **class-map** command when the map is shared among mulitple policies. When you enter the **class-map** command, the device enters the class-map configuration mode.

You can create a default class by using the **class class-default** policy-map configuration command. The default class is system-defined and cannot be configured. Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as default traffic.

Time-to-Live Classification

You can classify packets based on the ACL map. You can set Time-to_live (TTL) as a criterion in the ACL list and perform a TTL check on the incoming packet. The access control entry is used to check the IPv4 TTL to match the value on the incoming packet. The classified packet is either marked or policed based on the policy-map action. Queueing cannot be configured on this classification.

The following is an example of TTL classification:

```
policy-map TTL_MATCH
 class IPV4_TTL
  police rate 6000000000
  set dscp af23

ip access-list extended IPV4_TTL
 permit ip any any ttl eq 100
 permit tcp any any ttl ne 150
```

```
!
Device#show run class-map IPV4_TTL
class-map match-all IPV4_TTL
 match access-group name IPV4_TTL
!

Device#show policy-map interface hun1/0/47

HundredGigE1/0/47

  Service-policy output: TTL_MATCH

    Class-map: IPV4_TTL (match-all)
    553567424 packets
    Match: access-group name IPV4_TTL
    police:
    rate 6000000000 bps, burst 187500000 bytes
    conformed 22983406600 bytes; actions:
    transmit
    exceeded 32375773000 bytes; actions:
    drop
    conformed 588922000 bps, exceeded 830894000 bps
    QoS Set
    dscp af23

    Class-map: class-default (match-any)
    2184433710 packets
    Match: any
```

Layer 3 Packet Length Classification

This feature provides the capability of matching and classifying traffic on the basis of the Layer 3 packet length in the IP header. The Layer 3 packet length is the IP datagram length plus the IP header length. You can set the packet length as a matching criterion in the class policy-map, to match the value on the incoming packet. The classified packet is either marked or policed based on the policy-map action. This feature does not work on IPv6 packets.

The following is an example of Layer 3 packet length classification:

```
Service-policy output: PACKET_MATCH1

Class-map: class-default (match-any)
 16281588 packets
  Match: any

Service-policy : L3_MATCH

Class-map: PACKET_LENGTH_1 (match-any)
  9910510 packets
  Match: packet length 7582
  Match: packet length 5000
  QoS Set
  dscp cs2
  police:
 rate 3 %
 rate 1200000000 bps, burst 37500000 bytes
 conformed 10000 bytes; actions:
   transmit
 exceeded 112121 bytes; actions:
   drop
 conformed 500 bps, exceeded 3434 bps

Class-map: PACKET_LENGTH_2 (match-all)
```

```
 6371042 packets
  Match: dscp cs4 (32)
  Match: packet length 7759
  police:
 rate 12000000000 bps, burst 375000000 bytes
 conformed 44545 bytes; actions:
 transmit
 exceeded 34343 bytes; actions:
 drop
 conformed 1211 bps, exceeded 11211 bps

Class-map: class-default (match-any)
 36 packets
  Match: any
 QoS Set
   precedence 3
Device#

class-map match-any PACKET_LENGTH_1
match packet length min 7582 max 7582
match packet length min 5000 max 5000

class-map match-all PACKET_LENGTH_2
match dscp cs4
match packet length min 7759 max 7759
```

Layer 2 SRC-Miss or DST-Miss Classification

Traffic can be classified for a missing MAC address in the MAC address table, for source MAC address or destination MAC address. Policy-map with L2-Miss classification can be applied on layer 2 interfaces, in the ingress direction. Policing, marking or remarking actions can be applied with this classification. L2-Miss classification cannot be applied on layer 3 interfaces. Queueing cannot be configured on this classification.

The following is an example of L2-Miss classification:

```
Device #show run class-map DST-MISS
    class-map match-any DST-MISS
    match l2 dst-mac miss

Device #show run class-map SRC-MISS
    class-map match-all SRC-MISS
    match l2 src-mac miss

Device #show policy-map L2-MISS
Policy Map L2-MISS
    Class DST-MISS
        set dscp af22
    police cir percent 10
        conform-action transmit
        exceed-action drop
    Class SRC-MISS
        set precedence 1
    police rate percent 20
        conform-action transmit
        exceed-action drop
!
end

Device#
```

Policy Maps

A policy map specifies which traffic class to act on. Actions can include the following:

- Setting a specific DSCP or IP precedence value in the traffic class
- Setting a CoS value in the traffic class
- Setting a QoS group
- Specifying the traffic bandwidth limitations and the action to take when the traffic is out of profile

Before a policy map can be effective, you must attach it to a port.

You create and name a policy map using the **policy-map** global configuration command. When you enter this command, the device enters the policy-map configuration mode. In this mode, you specify the actions to take on a specific traffic class by using the **class** or **set** policy-map configuration and policy-map class configuration commands.

The policy map can also be configured using the **police** and **bandwidth** policy-map class configuration commands, which define the policer, the bandwidth limitations of the traffic, and the action to take if the limits are exceeded. In addition, the policy-map can further be configured using the **priority** policy-map class configuration command, to schedule priority for the class or the queuing policy-map class configuration commands, **queue-buffers** and **queue-limit**.

To enable the policy map, you attach it to a port by using the **service-policy** interface configuration command.

Note You cannot configure both **priority** and **set** for a policy map. If both these commands are configured for a policy map, and when the policy map is applied to an interface, error messages are displayed. The following example shows this restriction:

```
Device# configure terminal
Device(config)# class-map cmap
Device(config-cmap)# exit
Device(config)# class-map classmap1
Device(config-cmap)# exit
Device(config)# policy-map pmap
Device(config-pmap)# class cmap
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# exit
Device(config-pmap)# class classmap1
Device(config-pmap-c)# set dscp 10
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/2
Device(config-if)# service-policy output pmap

Non-queuing action only is unsupported in a queuing policy!!!
%QOS-6-POLICY_INST_FAILED:
Service policy installation failed
```

Policy Map on Physical Port

You can configure a nonhierarchical policy map on a physical port that specifies which traffic class to act on. Actions can include setting a specific DSCP or IP precedence or CoS values in the traffic class, specifying the traffic bandwidth limitations for each matched traffic class (policer), and taking action when the traffic is out of profile (marking).

A policy map also has these characteristics:

- A policy map can contain multiple class statements, each with different match criteria and policers.
- A policy map can contain a predefined default traffic class explicitly placed at the end of the map.

  When you configure a default traffic class by using the **class class-default** policy-map configuration command, unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as the default traffic class (**class-default**).
- A separate policy-map class can exist for each type of traffic received through a port.

Policy Map on VLAN

The device supports a VLAN QoS feature that allows the user to perform QoS treatment at the VLAN level (classification and QoS actions) using the incoming frame's VLAN information. In VLAN-based QoS, a service policy is applied to an SVI interface. All physical interfaces belonging to a VLAN policy map then need to be programmed to refer to the VLAN-based policy maps instead of the port-based policy map.

Although the policy map is applied to the VLAN SVI, only marking or remarking actions can be performed on a per-port basis. You cannot configure the policer to take account of the sum of traffic from a number of physical ports. Each port needs to have a separate policer governing the traffic coming into that port.

QoS Profile

The device uses Ternary Content Addressable Memory (TCAM) to store classification rules. To optimize the usage of TCAM resources, use the QoS profile to turn off some of the lesser used features and turn them on when required.

With the **qos profile** {**default** | **extended**} command, you can select the required classification feature set. **default** keyword loads only the common classification features. **extended** keyword loads the complete classification feature set (but with a reduced scale) that are available for the device. By default, only the commonly used classification features are set on the device.

The following are the extra features that are supported by the extended feature set:

- Time-to-Live
- Source-Miss and Destination-Miss
- TCP Flags

You can verify the QoS profile that is configured on the device using the **show platform software fed active qos profile** command.

Example

```
device# show platform software fed active qos profile
Using default - Common Classification Features
```

Security Group Classification

Security Group classification includes both source and destination groups, which are specified by source security group tag (SGT) and destination security group tag (DGT) respectively.

The objective of SGT QoS classification is to leverage user groups to increase policy granularity such that the policy isn't only application-aware but also provides some level of differentiated service based on the user identity (or the group of users to which they belong).

Egress QoS classification based on SGT or DGT isn't supported.

SGT Based QoS

The SGT based QoS feature provides a special treatment for a class of traffic that is based on the QoS policies and actions, for a defined user group or device. This feature enables you to assign multiple QoS policies to an application or traffic type that is initiated by different user groups. Each user group is defined by a unique SGT value and can support MQC-based QoS configuration.

The SGT based QoS feature is applicable to both the user group and the device-based QoS service levels for SGT-DGT-based packet classification. It can also potentially support defining of user groups based on contextual information for QoS policy prioritization.

Sharing DGID with SGACL

Due to resource limitations, only 4096 security group destination tags (DGTs) are supported. Classification based on DGT is achieved through a security destination tag ID known as DGID. DGID is a global resource and is shared with SGACL. DGID allocation is done on a first-come-first-serve basis. On a device, at startup, SGACL configuration is applied before QoS policy configuration. Hence DGID is first allocated for SGACL and then for QoS policy.

The **show platform software fed sw active sgacl detail** command displays the DGT to DGID mapping.

Example

```
device# show platform software fed active sgacl detail

Global Enforcement: On
*Refcnt: for the non-SGACL feature
========================== DGID Table ====================================
SGT/Refcnt     DGT     DGID   hash  test_cell monitor   permitted  denied
==========================================================================
*/1            24        1     24
 24            24        1     24       Off     Off        0           0
```

Restrictions for SGT Based QoS

The following are the limitations of the SGT based QoS feature:

- SGT based QoS is not supported on tunnel interfaces.
- Only 4096 security destination tags and 65539 security source tags are supported.
- SGT based policy can only be attached to the input direction of an interface.

Restrictions for an Upgrade or Downgrade

- For an upgrade from an earlier release to Cisco IOS XE Release 16.12.x and later, the maximum supported DGID is 256. Reload the switch to overcome this issue.
- For a downgrade from Cisco IOS XE Release 17.1.x to IOS XE 16.12.x releases, the allocated DGID is displayed as 4096; but only 256 DGIDs are supported. Reload the switch to overcome this issue.
- An ISSU upgrade fails if the tcp flag, time-to-live, source-miss, and destination-miss are set in a policy. To do an ISSU, first remove the tcp flag, TTL, source-miss, and destination-miss configurations.

- If a policy-map, which is attached to an interface, classifies traffic based on tcp flag, an ISSU upgrade fails. To do an ISSU, either detach the policy-map from the interface or remove the tcp flag classification.

Ingress Port FIFO Parser

Ingress Port FIFO (IPF) parses incoming network traffic to classify frames into different priorities levels. It derives the traffic class from different packet formats. For example, the traffic class can be derived from the Differentiated Services Code Point (DSCP) for IP packets, or, Class of Service (CoS) for dot1q tag packets. These traffic classes are further mapped to priority levels, which are used to take drop decisions, in case of congestion.

The IPF parser, can be used in the global mode and in the isolation mode (high and low priority configuration at port level). By default it is in the isolation mode. In the isolation mode, priority differentiation is made at the port level rather than the system level.

To configure the IPF parser in global configuration mode, use the following command:

```
Device(config)# qos port-ingress-fifo mode global
```

The following are the examples of **show** commands to see the traffic class to priority mappings:

```
Device# show platform hardware fed active qos ipf interface twentyFiveGigE 1/0/1 cos-map
IPF cos to traffic class map for Interface [cos : traffic-class]:
------------------------------------------------------------------------
0 : 0           1 : 1           2 : 2           3 : 3
4 : 4           5 : 5           6 : 6           7 : 7
8 : 4           9 : 4           10 : 4          11 : 4
12 : 4          13 : 4          14 : 4          15 : 4
```

```
Device# show platform hardware fed active qos ipf interface twentyFiveGigE 1/0/1 dscp-map
IPF dscp to traffic class map for Interface [dscp : traffic-class]:
----------------------------------------------------------------
0 : 0           1 : 0           2 : 0           3 : 0
4 : 0           5 : 0           6 : 0           7 : 0
8 : 1           9 : 1           10 : 1          11 : 1
12 : 1          13 : 1          14 : 1          15 : 1
16 : 2          17 : 4          18 : 4          19 : 4
20 : 4          21 : 4          22 : 4          23 : 4
24 : 3          25 : 4          26 : 4          27 : 4
28 : 4          29 : 4          30 : 4          31 : 4
32 : 4          33 : 4          34 : 4          35 : 4
36 : 4          37 : 4          38 : 4          39 : 4
40 : 4          41 : 4          42 : 4          43 : 4
44 : 4          45 : 4          46 : 5          47 : 4
48 : 6          49 : 4          50 : 4          51 : 4
52 : 4          53 : 4          54 : 4          55 : 4
56 : 7          57 : 4          58 : 4          59 : 4
60 : 4          61 : 4          62 : 4          63 : 4
```

```
Device#show platform hardware fed active qos ipf interface twentyFiveGigE 1/0/1 exp-map
IPF exp to traffic class map for Interface [exp : traffic-class]:
----------------------------------------------------------------
0 : 0           1 : 1           2 : 2           3 : 3
4 : 4           5 : 5           6 : 6           7 : 7
```

```
Device#show platform hardware qos ipf interface twentyFiveGigE 1/0/1 ipf-parse-cfg
IPF configration for Interface:
-------------------------------
Port Trust:           Enabled
Default TC:           0
Dscp based parsing:   Disabled
Exp based parsing:    Disabled
```

```
Fdcos based parsing:  Enabled
cos based parsing:    Disabled

Device#show platform hardware fed active qos ipf tc-to-pri asic 0
IPF traffic class to priority for[Asic:Core:TlaInst]::[0:0:0]
------------------------------------------------------------
Priority               Traffic Classes
--------               ---------------
Low Pri :              0  1  4
High Pri:              2  3  5  6  7
IPF traffic class to priority for[Asic:Core:TlaInst]::[0:0:1]
------------------------------------------------------------
Priority               Traffic Classes
--------               ---------------
Low Pri :              0  1  4
High Pri:              2  3  5  6  7
```

Statistics show command:

```
Device#show platform hardware fed active qos ipf statistics asic 0
Ipf Statistics:[Asic|Core|Tla] : [0 | 0 | 0] - Global Mode
-----------------------------------------------
Ipf misc packet drops:                          0
Ipf Drop Statistics
-------------------
low pri Frames drop:                      0
low pri mop Frames drop:                  0
high pri Frames drop:                     0
almost full Frames drop:                  0
RCP Frames drop:                          0

Ipf Statistics:[Asic|Core|Tla] : [0 | 0 | 1] - Global Mode
-----------------------------------------------
Ipf misc packet drops:                          0
Ipf Drop Statistics
-------------------
low pri Frames drop:                      0
low pri mop Frames drop:                  0
high pri Frames drop:                     0
almost full Frames drop:                  0
RCP Frames drop:                          0
```

Policing

After a packet is classified and has a DSCP-based, CoS-based, or QoS-group label assigned to it, the policing and marking process can begin.

Policing involves creating a policer that specifies the bandwidth limits for the traffic. Packets that exceed the limits are *out of profile* or *nonconforming*. Each policer decides on a packet-by-packet basis whether the packet is in or out of profile and specifies the actions on the packet. These actions, carried out by the marker, include passing through the packet without modification, dropping the packet, or modifying (marking down) the assigned DSCP or CoS value of the packet and allowing the packet to pass through.

To avoid out-of-order packets, both conform and nonconforming traffic typically exit the same queue.

Note All traffic, regardless of whether it is bridged or routed, is subjected to a policer, if one is configured. As a result, bridged packets might be dropped or might have their DSCP or CoS fields modified when they are policed and marked.

You can only configure policing on a physical port.

After you configure the policy map and policing actions, attach the policy-map to an ingress or egress port by using the **service-policy** interface configuration command.

Token-Bucket Algorithm

Policing uses a token-bucket algorithm. As each frame is received by the device, a token is added to the bucket. The bucket has a hole in it and leaks at a rate that you specify as the average traffic rate in bits per second. Each time a token is added to the bucket, the device verifies that there is enough room in the bucket. If there is not enough room, the packet is marked as nonconforming, and the specified policer action is taken (dropped or marked down).

How quickly the bucket fills is a function of the bucket depth (burst-byte), the rate at which the tokens are removed (rate-bps), and the duration of the burst above the average rate. The size of the bucket imposes an upper limit on the burst length and limits the number of frames that can be transmitted back-to-back. If the burst is short, the bucket does not overflow, and no action is taken against the traffic flow. However, if a burst is long and at a higher rate, the bucket overflows, and the policing actions are taken against the frames in that burst.

You configure the bucket depth (the maximum burst that is tolerated before the bucket overflows) by using the burst-byte option of the **police** policy-map class configuration command. You configure how fast (the average rate) that the tokens are removed from the bucket by using the rate option of the **police** policy-map class configuration command.

Marking

Marking is used to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another.

Marking can be used to set certain field/bits in the packet headers, or marking can also be used to set certain fields in the packet structure that is internal to the device. Additionally, the marking feature can be used to define mapping between fields. The following marking methods are available for QoS:

- Packet header
- Device specific information
- Table maps

Packet Header Marking

Marking on fields in the packet header can be classified into two general categories:

- IPv4/v6 header bit marking
- Layer 2 header bit marking

The marking feature at the IP level is used to set the precedence or the DSCP in the IP header to a specific value to get a specific per-hop behavior at the downstream device (switch or router), or it can also be used to aggregate traffic from different input interfaces into a single class in the output interface. The functionality is currently supported on both the IPv4 and IPv6 headers.

Marking in the Layer 2 headers is typically used to influence dropping behavior in the downstream devices (switch or router). It works in tandem with the match on the Layer 2 headers. The bits in the Layer 2 header that can be set using a policy map are class of service.

Switch-Specific Information Marking

This form of marking includes marking of fields in the packet data structure that are not part of the packets header, so that the marking can be used later in the data path. This is not propagated between the switches. Marking of QoS group falls into this category. This form of marking is only supported in policies that are enabled on the input interfaces. The corresponding matching mechanism can be enabled on the output interfaces on the same switch and an appropriate QoS action can be applied.

Table Map Marking

Table map marking enables the mapping and conversion from one field to another using a conversion table. This conversion table is called a table map.

Depending upon the table map attached to an interface, CoS, DSCP, and Precedence values of the packet are rewritten. The device allows configuring both ingress table map policies and egress table map policies.

As an example, a table map can be used to map the Layer 2 CoS setting to a precedence value in Layer 3. This feature enables combining multiple **set** commands into a single table, which indicates the method to perform the mapping. This table can be referenced in multiple policies, or multiple times in the same policy.

A table map-based policy supports the following capabilities:

- Mutation: You can have a table map that maps from one DSCP value set to another DSCP value set, and this can be attached to an egress port.
- Rewrite: Packets coming in are rewritten depending upon the configured table map.
- Mapping: Table map based policies can be used instead of set policies.

The following steps are required for table map marking:

1. Define the table map: Use the **table-map** global configuration command to map the values. The table does not know of the policies or classes within which it will be used. The default command in the table map is used to indicate the value to be copied into the to field when there is no matching from field.
2. Define the policy map: You must define the policy map where the table map will be used.
3. Associate the policy to an interface.

Note A table map policy on an input port changes the trust setting of that port to the from type of qos-marking.

Note In order to trust a value other than the dscp value, use table map with default copy in the ingress direction.

Note When you map a QoS group to a DSCP value in an egress table map policy, the QoS group does not map the equivalent COS value for DSCP. Configure a separate QoS group to COS table map if you want to define the QoS group to a non-zero COS value.

Traffic Conditioning

To support QoS in a network, traffic entering the service provider network needs to be policed on the network boundary routers to ensure that the traffic rate stays within the service limit. Even if a few routers at the network boundary start sending more traffic than what the network core is provisioned to handle, the increased traffic load leads to network congestion. The degraded performance in the network makes it difficult to deliver QoS for all the network traffic.

Traffic policing functions (using the police feature) and shaping functions (using the traffic shaping feature) manage the traffic rate, but differ in how they treat traffic when tokens are exhausted. The concept of tokens comes from the token bucket scheme, a traffic metering function.

Note When running QoS tests on network traffic, you may see different results for the shaper and policing data. Network traffic data from shaping provides more accurate results.

This table compares the policing and shaping functions.

Table 4: Comparison Between Policing and Shaping Functions

Policing Function	Shaping Function
Sends conforming traffic up to the line rate and allows bursts.	Smooths traffic and sends it out at a constant rate.
When tokens are exhausted, action is taken immediately.	When tokens are exhausted, it buffers packets and sends them out later, when tokens are available. A class with shaping has a queue associated with it which will be used to buffer the packets.
Policing has multiple units of configuration – in bits per second, packets per second and cells per second.	Shaping has only one unit of configuration - in bits per second.
Policing has multiple possible actions associated with an event, marking and dropping being example of such actions.	Shaping does not have the provision to mark packets that do not meet the profile.
Works for both input and output traffic.	Implemented for output traffic only.
Transmission Control Protocol (TCP) detects the line at line speed but adapts to the configured rate when a packet drop occurs by lowering its window size.	TCP can detect that it has a lower speed line and adapt its retransmission timer accordingly. This results in less scope of retransmissions and is TCP-friendly.

Policing

The QoS policing feature is used to impose a maximum rate on a traffic class. The QoS policing feature can also be used with the priority feature to restrict priority traffic. If the rate is exceeded, then a specific action is taken as soon as the event occurs. The rate (committed information rate [CIR] and peak information rate [PIR]) and the burst parameters (conformed burst size [B_c] and extended burst size [B_e]) are all configured in bytes per second.

The following policing forms or policers are supported for QoS:

- Single-rate two-color policing
- Dual-rate three-color policing

Note Single-rate three-color policing is not supported.

Single-Rate Two-Color Policing

Single-rate two-color policer is the mode in which you configure only a CIR and a B_c.

The B_c is an optional parameter, and if it is not specified it is computed by default. In this mode, when an incoming packet has enough tokens available, the packet is considered to be conforming. If at the time of packet arrival, enough tokens are not available within the bounds of B_c, the packet is considered to have exceeded the configured rate.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 65.

Dual-Rate Three-Color Policing

With the dual rate policer, the device supports only color-blind mode. In this mode, you configure a committed information rate (CIR) and a peak information rate (PIR). As the name suggests, there are two token buckets in this case, one for the peak rate, and one for the conformed rate.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 65.

In the color-blind mode, the incoming packet is first checked against the peak rate bucket. If there are not enough tokens available, the packet is said to violate the rate. If there are enough tokens available, then the tokens in the conformed rate buckets are checked to determine if there are enough tokens available. The tokens in the peak rate bucket are decremented by the size of the packet. If it does not have enough tokens available, the packet is said to have exceeded the configured rate. If there are enough tokens available, then the packet is said to conform, and the tokens in both the buckets are decremented by the size of the packet.

The rate at which tokens are replenished depends on the packet arrival. Assume that a packet comes in at time T1 and the next one comes in at time T2. The time interval between T1 and T2 determines the number of tokens that need to be added to the token bucket. This is calculated as:

Time interval between packets (T2-T1) * CIR)/8 bytes

Shaping

Shaping is the process of imposing a maximum rate of traffic, while regulating the traffic rate in such a way that the downstream switches and routers are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface.

Shaping has a buffer associated with it that ensures that packets which do not have enough tokens are buffered as opposed to being immediately dropped. The number of buffers available to the subset of traffic being shaped is limited and is computed based on a variety of factors. The number of buffers available can also be tuned using specific QoS commands. Packets are buffered as buffers are available, beyond which they are dropped.

Class-Based Traffic Shaping

The device uses class-based traffic shaping. This shaping feature is enabled on a class in a policy that is associated to an interface. A class that has shaping configured is allocated a number of buffers to hold the packets that do not have tokens. The buffered packets are sent out from the class using FIFO. In the most common form of usage, class-based shaping is used to impose a maximum rate for an physical interface or logical interface as a whole. The following shaping forms are supported in a class:

- Average rate shaping
- Hierarchical shaping

Shaping is implemented using a token bucket. The values of CIR, B_c and B_e determine the rate at which the packets are sent out and the rate at which the tokens are replenished.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 65.

Average Rate Shaping

You use the **shape average** policy-map class command to configure average rate shaping.

This command configures a maximum bandwidth for a particular class. The queue bandwidth is restricted to this value even though the port has more bandwidth available. The device supports configuring shape average by either a percentage or by a target bit rate value.

Hierarchical Shaping

Shaping can also be configured at multiple levels in a hierarchy. This is accomplished by creating a parent policy with shaping configured, and then attaching child policies with additional shaping configurations to the parent policy.

The supported hierarchical shaping type is Port Shaper.

The port shaper uses the class default and the only action permitted in the parent is shaping. The queuing action is in the child with the port shaper. With the user configured shaping, you cannot have queuing action in the child.

Queuing and Scheduling

The device uses both queuing and scheduling to help prevent traffic congestion. The device supports the following queuing and scheduling features:

- Bandwidth
- Weighted Tail Drop
- Priority queues
- Queue buffers
- Weighted Random Early Detection

When you define a queuing policy on a port, control packets are mapped to the best priority queue with the highest threshold. Control packets queue mapping works differently in the following scenarios:

- Without a quality of service (QoS) policy: If no QoS policy is configured, control packets with DSCP values 16, 24, 48, and 56 are mapped to queue 0 with the highest threshold of threshold2.
- With an user-defined policy: An user-defined queuing policy configured on egress ports can affect the default priority queue setting on control packets.

Note Queuing policy in egress direction does not support **match access-group** classification.

Control traffic is redirected to the best queue based on the following rules:

1. If defined in a user policy, the highest- level priority queue is always chosen as the best queue.
2. In the absence of a priority queue, Cisco IOS software selects queue 0 as the best queue. When the software selects queue 0 as the best queue, you must define the highest bandwidth to this queue to get the best QoS treatment to the control plane traffic.
3. If thresholds are not configured on the best queue, Cisco IOS software assigns control packets with Differentiated Services Code Point (DSCP) values 16, 24, 48, and 56 are mapped to threshold2 and reassigns the rest of the control traffic in the best queue to threshold1.

If a policy is not configured explicitly for control traffic, the Cisco IOS software maps all unmatched control traffic to the best queue with threshold2, and the matched control traffic is mapped to the queue as configured in the policy.

Note To provide proper QoS for Layer 3 packets, you must ensure that packets are explicitly classified into appropriate queues. When the software detects DSCP values in the default queue, then it automatically reassigns this queue as the best queue.

Bandwidth

The device supports the following bandwidth configurations:

- Bandwidth
- Bandwidth percent
- Bandwidth remaining precent

Bandwidth Percent

You can use the **bandwidth percent** policy-map class command to allocate a minimum bandwidth to a particular class. The total sum cannot exceed 100 percent and in case the total sum is less than 100 percent, then the rest of the bandwidth is divided equally among all bandwidth queues.

Note A queue can oversubscribe bandwidth in case the other queues do not utilize the entire port bandwidth.

You cannot mix bandwidth types on a policy map. For example, you cannot configure bandwidth in a single policy map using both a bandwidth percent and in kilobits per second.

Bandwidth Remaining Percent

Use the **bandwidth remaining percent** policy-map class command to create a percent for sharing unused bandwidth in specified queues. Any unused bandwidth will be used by these specific queues in the percent that is specified by the configuration. Use this command when the **priority** command is also used for certain queues in the policy.

When you assign percent, the queues will be assigned certain weights which are inline with these percents.

You can specify a percent between 0 to 100. For example, you can configure a bandwidth remaining percent of 2 on one class, and another queue with a bandwidth remaining percent of 4 on another class. The bandwidth remaining percent of 4 will be scheduled twice as often as the bandwidth remaining percent of 2.

The total bandwidth percent allocation for the policy can exceed 100. For example, you can configure a queue with a bandwidth remaining percent of 50, and another queue with a bandwidth remaining precent of 100.

Weighted Tail Drop

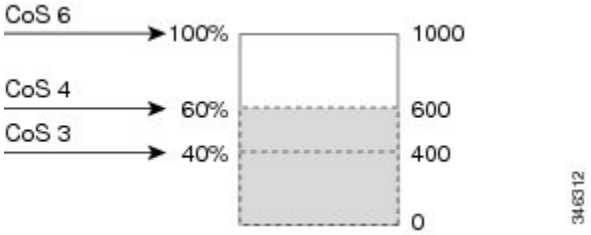
The device egress queues use an enhanced version of the tail-drop congestion-avoidance mechanism called weighted tail drop (WTD). WTD is implemented on queues to manage the queue lengths and to provide drop precedences for different traffic classifications.

As a frame is enqueued to a particular queue, WTD uses the frame's assigned QoS label to subject it to different thresholds. If the threshold is exceeded for that QoS label (the space available in the destination queue is less than the size of the frame), the device drops the frame.

Each queue has three configurable threshold values. The QoS label determines which of the three threshold values is subjected to the frame.

Figure 2: WTD and Queue Operation

The following figure shows an example of WTD operating on a queue whose size is 1000 frames. Three drop percentages are configured: 40 percent (400 frames), 60 percent (600 frames), and 100 percent (1000 frames). These percentages indicate that up to 400 frames can be queued at the 40-percent threshold, up to 600 frames at the 60-percent threshold, and up to 1000 frames at the 100-percent threshold.



In the example, CoS value 6 has a greater importance than the other CoS values, and is assigned to the 100-percent drop threshold (queue-full state). CoS values 4 is assigned to the 60-percent threshold, and CoS values 3 is assigned to the 40-percent threshold. All of these threshold values are assigned using the **queue-limit cos** command.

Assuming the queue is already filled with 600 frames, and a new frame arrives. It contains CoS value 4 and is subjected to the 60-percent threshold. If this frame is added to the queue, the threshold will be exceeded, so the device drops it.

Weighted Tail Drop Default Values

The following are the Weighted Tail Drop (WTD) default values and the rules for configuring WTD threshold values.

- If you configure less than three queue-limit percentages for WTD, then WTD default values are assigned to these thresholds.

  The following are the WTD threshold default values:

  Table 5: WTD Threshold Default Values

Threshold	Default Value Percentage
0	80
1	90
2	400

- If 3 different WTD thresholds are configured, then the queues are programmed as configured.
- If 2 WTD thresholds are configured, then the maximum value percentage will be 400.
- If a WTD single threshold is configured as x, then the maximum value percentage will be 400.
  - If the value of x is less than 90, then threshold1=90 and threshold 0= x.
  - If the value of x equals 90, then threshold1=90, threshold 0=80.
  - If the value x is greater than 90, then threshold1=x, threshold 0=80.

Priority Queues

Each port supports eight egress queues, of which two can be given a priority.

You use the **priority level** policy class-map command to configure the priority for two classes. One of the classes has to be configured with a priority queue level 1, and the other class has to be configured with a priority queue level 2. Packets on these two queues are subjected to less latency with respect to other queues.

You cannot send 100 percent line rate traffic when a priority queue is configured. There can only be 99.6 percent line rate traffic with priority queue configured, ensuring a latency of less than 20 microseconds.

Note You can configure a priority only with a level.

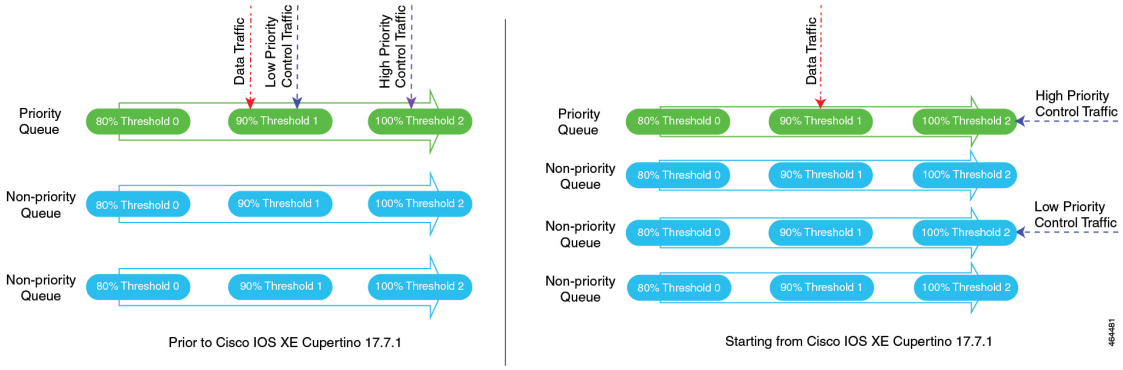
Strict priority is allowed with priority level 1 and priority level 2, in one policy map.

Prior to Cisco IOS XE Cupertino 17.7.1, system generated low-priority CPU traffic is mapped to threshold 1 of a priority queue as seen in the Figure.

Beginning from Cisco IOS XE Cupertino 17.7.1, system generated low-priority CPU traffic is mapped to threshold 2 of a non-priority queue with highest bandwidth. For example, in the Figure, non-priority CPU traffic such as BGP traffic is mapped to threshold 2 of a non-priority queue with highest bandwidth.

Note If there is an issue between system generated traffic and user data traffic for high bandwidth queues, then you can use the weighted-tail drop configurations to map the user data traffic to a different threshold.

Figure 3: Traffic Mapping to a Priority Queue



Priority Queue Policer

The switch supports configuration of policing rate on priority queue. Priority queue policer supports only a single-rate two-color policing.

Note Policing with table-maps is not supported.

Examples of Configuring Priority Queue Policer

Example 1

```
Policy Map priority-1
    Class prio1
      priority level 1
     police rate percent 10
       conform-action transmit
       exceed-action drop
    Class prio2
      priority level 2
     police rate percent 5
       conform-action transmit
       exceed-action drop
    Class new
      bandwidth 20 (%)
```

Example 2

```
Policy Map priority-1
    Class prio1
      priority level 1 20 (%)
     police rate percent 10
       conform-action transmit
       exceed-action drop
    Class prio2
      priority level 2 25 (%)
     police rate percent 5
       conform-action transmit
       exceed-action drop
    Class new
      bandwidth 20 (%)
```

Queue Buffer

At boot time, when there is no policy map enabled on the wired port, there are two queues created by default. Wired ports can have a maximum of 8 queues configured using MQC-based policies. The following table shows which packets go into which one of the queues:

Table 6: Queue Threshold Mapping Table for DSCP, Precedence, and CoS

DSCP, Precedence or CoS	Queue	Threshold
Control Packets	0	2
Rest of Packets	1	2

Note You can guarantee the availability of buffers, set drop thresholds, and configure the maximum memory allocation for a queue. You use the **queue-buffers** policy-map class command to configure the queue buffers. You use the **queue-limit** policy-map class command to configure the maximum thresholds.

There are two types of buffer allocations: hard buffers, which are explicitly reserved for the queue, and soft buffers, which are available for other ports when unused by a given port.

For the wired port default, Queue 0 will be given 40 percent of the buffers that are available for the interface as hard buffers, that is 200 buffers are allocated for Queue 0 in the context of 1-gigabit ports, and 1200 buffers in the context of 10-gigabit ports. The soft maximum for this queue is set to four times the hard buffer, which is 800 for 1-gigabit ports and 2400 for 10-gigabit ports, and 19200 for 40-gigabit ports, where 400 is the default maximum threshold that is configured for any queue.

Queue 1 does not have any hard buffers allocated. Soft buffers have a minimum allocation of 300 buffers for 1-gigabit ports, 1800 buffers for 10-gigabit ports and 7200 buffers for 40-gigabit ports. The soft maximum allocation for Queue 1 is four times the soft minimum with 1200 buffers for 1-gigabit ports, 7200 buffers for 10-gigabit ports and 28800 buffers for 40-gigabit ports.

Note By default, Queue 0 is not a priority queue. A policy-map can enable Queue 0 to be a priority queue by using the **priority level** command. If Queue 0 is assigned a priority level of 1, the soft maximum limit for this queue is automatically set to the same value as the hard maximum limit.

Queue Buffer Allocation

The buffer allocation to any queue can be tuned using the **queue-buffers ratio** policy-map class configuration command.

Dynamic Threshold and Scaling

Traditionally, reserved buffers are statically allocated for each queue. No matter whether the queue is active or not, its buffers are held up by the queue. In addition, as the number of queues increases, the portion of the reserved buffers allocated for each queue can become smaller and smaller. Eventually, a situation may occur where there are not enough reserved buffers to support a jumbo frame for all queues.

The device supports Dynamic Thresholding and Scaling (DTS), which is a feature that provides a fair and efficient allocation of buffer resources. When congestion occurs, this DTS mechanism provides an elastic buffer allocation for the incoming data based on the occupancy of the global/port resources. Conceptually, DTS scales down the queue buffer allocation gradually as the resources are used up to leave room for other queues, and vice versa. This flexible method allows the buffers to be more efficiently and fairly utilized.

As mentioned in the previous sections, there are two limits configured on a queue—a hard limit and a soft limit.

Hard limits are not part of DTS. These buffers are available only for that queue. The sum of the hard limits should be less than the globally set up hard maximum limit. The global hard limit configured for egress queuing is currently set to 5705. In the default scenario when there are no MQC policies configured, the 24 1-gigabit ports would take up 24 * 67 = 1608, and the 4 10-gigabit ports would take up 4 * 720 = 2880, for a total of 4488 buffers, allowing room for more hard buffers to be allocated based upon the configuration.

Soft limit buffers participate in the DTS process. Additionally, some of the soft buffer allocations can exceed the global soft limit allocation. The global soft limit allocation for egress queuing is currently set to 27024. The sum of the hard and soft limits add up to 39696, which in turn translates to 10.1 MB. Because the sum of the soft buffer allocations can exceed the global limit, it allows a specific queue to use a large number of buffers when the system is lightly loaded. The DTS process dynamically adjusts the per-queue allocation as the system becomes more heavily loaded.

Unified Buffer Sharing

Starting with the Cisco IOS XE 17.2.1 release, you can configure sharing of Active Queue Management (AQM) buffers between the two cores inside the same ASIC. A port configured with buffer sharing will be able to use any of the available AQM buffers regardless of the cores to which the AQM buffers are mapped. This will help manage higher bursts of traffic that would have saturated the buffer of a single AQM core.

You can enable this feature by using the **qos share-buffer** command. You can check if buffer sharing has been enabled using the **show plat hardware fed active qos queue config interface** command. This will be a global configuration that will affect the whole system. You can disable buffer sharing by using the **no** form of the command, **no qos share-buffer**.

Weighted Random Early Detection

Weighted random early detection (WRED) is a mechanism to avoid congestion in networks. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion, thus avoiding large number of packet drops at once.

For more information about WRED, see Configuring Weighted Random Early Detection, on page 227.

Trust Behavior

The following sections provide information about trust behavior.

Port Security on a Trusted Boundary for Cisco IP Phones

In a typical network, you connect a Cisco IP Phone to a device port and cascade devices that generate data packets from the back of the telephone. The Cisco IP Phone guarantees the voice quality through a shared data link by marking the CoS level of the voice packets as high priority (CoS = 5) and by marking the data packets as low priority (CoS = 0). Traffic sent from the telephone to the device is typically marked with a tag that uses the 802.1Q header. The header contains the VLAN information and the class of service (CoS) 3-bit field, which is the priority of the packet.

For most Cisco IP Phone configurations, the traffic sent from the telephone to the device should be trusted to ensure that voice traffic is properly prioritized over other types of traffic in the network. By using the **trust device** interface configuration command, you configure the device port to which the telephone is connected to trust the traffic received on that port.

Note The **trust device** *device_type* command available in interface configuration mode is a stand-alone command on the device. When using this command in an AutoQoS configuration, if the connected peer device is not a corresponding device (defined as a device matching your trust policy), both CoS and DSCP values are set to "0" and any input policy will not take effect. If the connected peer device is a corresponding device, input policy will take effect.

With the trusted setting, you also can use the trusted boundary feature to prevent misuse of a high-priority queue if a user bypasses the telephone and connects the PC directly to the device. Without trusted boundary, the CoS labels generated by the PC are trusted by the device (because of the trusted CoS setting). By contrast, trusted boundary uses CDP to detect the presence of a Cisco IP Phone (such as the Cisco IP Phone 7910, 7935, 7940, and 7960) on a device port. If the telephone is not detected, the trusted boundary feature disables the trusted setting on the device port and prevents misuse of a high-priority queue. Note that the trusted boundary feature is not effective if the PC and Cisco IP Phone are connected to a hub that is connected to the device.

Trust Behavior for Wired Ports

In scenarios where the incoming packet type differs from the outgoing packet type, the trust behavior and the queuing behavior are explained in the following table. Note that the default trust mode for a port is DSCP based. The trust mode ‘falls back’ to CoS if the incoming packet is a pure Layer 2 packet. You can also change the trust setting from DSCP to CoS. This setting change is accomplished by using an MQC policy that has a class default with a 'set cos cos table default default-cos' action, where default-cos is the name of the table map created (which only performs a default copy).

For wired ports that are connected to the device (end points such as IP phones, laptops, cameras, telepresence units, or other devices), the trust device configuration is enabled on the interface. Their DSCP, precedence, or CoS values coming in from these end points are trusted by the device and therefore are retained in the absence of any explicit policy configuration.

The packets are enqueued to the appropriate queue per the default initial configuration. No priority queuing at the device is done by default. This is true for unicast and multicast packets.

Table 7: Trust and Queuing Behavior

Incoming Packet	Outgoing Packet	Trust Behavior	Queuing Behavior
Layer 3	Layer 3	Preserve DSCP/Precedence	Based on DSCP
Layer 2	Layer 2	Not applicable	Based on CoS
Tagged	Tagged	Preserve DSCP and CoS	Based on DSCP (trust DSCP takes precedence)
Layer 3	Tagged	Preserve DSCP, CoS is set to 0	Based on DSCP

Default Wired QoS Configuration

There are two queues configured by default on each wired interface on the device. All control traffic traverses and is processed through queue 0. All other traffic traverses and is processed through queue 1.

DSCP Maps

This section provides information about DSCP maps.

Default CoS-to-DSCP Map

When DSCP transparency mode is disabled, the DSCP values are derived from CoS as per the following table. If these values are not appropriate for your network, you need to modify them.

Note The DSCP transparency mode is disabled by default. If it is enabled (**no qos rewrite ip dscp** configuration command), DSCP rewrite will not happen.

Table 8: Default CoS-to-DSCP Map

CoS Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Default IP-Precedence-to-DSCP Map

You use the IP-precedence-to-DSCP map to map IP precedence values in incoming packets to a DSCP value that QoS uses internally to represent the priority of the traffic. The following table shows the default IP-precedence-to-DSCP map. If these values are not appropriate for your network, you need to modify them.

Table 9: Default IP-Precedence-to-DSCP Map

IP Precedence Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Default DSCP-to-CoS Map

You use the DSCP-to-CoS map to generate a CoS value, which is used to select one of the four egress queues. The following table shows the default DSCP-to-CoS map. If these values are not appropriate for your network, you need to modify them.

Table 10: Default DSCP-to-CoS Map

DSCP Value	CoS Value
0–7	0
8–15	1
16–23	2
24–31	3
32–39	4
40–47	5
48–55	6
56–63	7

How to Configure QoS

How to Configure Class, Policy, and Maps

The following sections provide configuration information about class, policy, and maps.

Creating a Traffic Class

To create a traffic class containing match criteria, use the **class-map** command to specify the traffic class name, and then use the following **match** commands in class-map configuration mode, as needed.

Before you begin

All match commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** *class-map name* {**match-any** | **match-all**}
	Example:
	`Device(config)# class-map test_1000`
	`Device(config-cmap)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• **match-any:** Any one of the match criteria must be met for traffic entering the traffic class to be classified as part of it.
		• **match-all:** All of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** {*index number* | *name*} **Example:** `Device(config-cmap)# match access-group`	The following parameters are available for this command: • access-group

	Command or Action	Purpose
	`100` `Device(config-cmap)#`	• cos • dscp • group-object • ip • mpls • precedence • protocol • qos-group • vlan • wlan (Optional) For this example, enter the access-group ID: • Access list index (value from 1 to 2799) • Named access list

Step 4	**match cos** *cos value*
	Example:
	`Device(config-cmap)# match cos 2 3 4 5`
	`Device(config-cmap)#`
		(Optional) Matches IEEE 802.1Q or ISL class of service (user) priority values.
		• Enters up to 4 CoS values separated by spaces (0 to 7).

Step 5	**match dscp** *dscp value*
	Example:
	`Device(config-cmap)# match dscp af11 af12`
	`Device(config-cmap)#`
		(Optional) Matches the DSCP values in IPv4 and IPv6 packets.

Step 6	**match ip** { **dscp** *dscp value* | **precedence** *precedence value* }
	Example:
	`Device(config-cmap)# match ip dscp af11 af12`
	`Device(config-cmap)#`
		(Optional) Matches IP values including the following:
		• **dscp**: Matches IP DSCP (DiffServ codepoints).
		• **precedence**: Matches IP precedence (0 to 7).
		Note Since CPU generated packets are not marked at egress, the packet will not match the configured class-map.

	Command or Action	Purpose

Step 7	**match qos-group** *qos group value*
	Example:
	`Device(config-cmap)# match qos-group 10`
	`Device(config-cmap)#`
		(Optional) Matches QoS group value (from 0 to 31).

Step 8	**match vlan** *vlan value*
	Example:
	`Device(config-cmap)# match vlan 210`
	`Device(config-cmap)#`
		(Optional) Matches a VLAN ID (from 1 to 4095).

Step 9	**end** **Example:** `Device(config-cmap)# end`	Saves the configuration changes.

What to do next

Configure the policy map.

Creating a Traffic Policy

To create a traffic policy, use the **policy-map** global configuration command to specify the traffic policy name.

The traffic class is associated with the traffic policy when the **class** command is used. The **class** command must be entered after you enter the policy map configuration mode. After entering the **class** command, the device is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

The following policy map class-actions are supported:

- bandwidth: Bandwidth configuration options.
- exit: Exits from the QoS class action configuration mode.
- no: Negates or sets default values for the command.
- police: Policer configuration options.
- priority: Strict scheduling priority configuration options for this class.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), this option is supported only in egress and queuing, but not in policy map.

- queue-buffers: Queue buffer configuration options.

- queue-limit: Queue maximum threshold for Weighted Tail Drop (WTD) configuration options.
- service-policy: Configures the QoS service policy.
- set: Sets QoS values using the following options:
  - CoS values
  - DSCP values
  - Precedence values
  - QoS group values
- shape: Traffic-shaping configuration options.

Note This option is not supported on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Before you begin

You should have first created a class map.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy-map name*
	Example:
	`Device(config)# policy-map test_2000`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** {*class-name* | **class-default**}
	Example:
	`Device(config-pmap)# class test_1000`
	`Device(config-pmap-c)#`
		Specifies the name of the class whose policy you want to create or change.
		You can also create a system default class for unclassified packets.

Step 4	**bandwidth** { *k* | **percent** *percentage* | **remaining** {*percent* | *ratio*} } **Example:**	(Optional) Sets the bandwidth using one of the following: • *Kb/s*: Kilobits per second, enter a value between 100 and 100000000 for Kb/s.

	Command or Action	Purpose
	`Device(config-pmap-c)# bandwidth 500` `Device(config-pmap-c)#`	• **percent**: Enter the percentage of the total bandwidth to be used for this policy map. • **remaining**: Enter the percentage ratio of the remaining bandwidth. **Note** On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), only the **bandwidth remaining** *ratio* option is supported. For a more detailed example of this command and its usage, see Configuring Bandwidth, on page 100.

Step 5	**exit**
	Example:
	`Device(config-pmap-c)# exit`
	`Device(config-pmap-c)#`
		(Optional) Exits from QoS class action configuration mode.

Step 6	**no**
	Example:
	`Device(config-pmap-c)# no`
	`Device(config-pmap-c)#`
		(Optional) Negates the command.

Step 7	**police** {*target_bit_rate* | **cir** | **rate**}
	Example:
	`Device(config-pmap-c)# police 100000`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*: Enter the bit rate per second, enter a value between 8000 and 10000000000.
		• **cir**: Committed Information Rate
		• **rate**: Specify police rate, PCR for hierarchical policies or SCR for single-level ATM 4.0 policer policies.
		For a more detailed example of this command and its usage, see Configuring Police , on page 102.

Step 8	**priority** {*kb/s* | **level** *level value* | **percent** *percentage value*} **Example:** `Device(config-pmap-c)# priority level`	(Optional) Sets the strict scheduling priority for this class. Command options include: • *kb/s*: Kilobits per second, enter a value between 1 and 2000000.

	Command or Action	Purpose
	`1 percent 50` `Device(config-pmap-c)#`	• **level**: Establishes a multi-level priority queue. Enter a value (1 or 2). • **percent**: Enter a percent of the total bandwidth for this priority. For a more detailed example of this command and its usage, see Configuring Priority, on page 104.

Step 9	**queue-buffers ratio** *ratio limit*
	Example:
	`Device(config-pmap-c)# queue-buffers ratio 10`
	`Device(config-pmap-c)#`
		(Optional) Configures the queue buffer for the class. Enter the queue buffers ratio limit (0 to 100).
		For a more detailed example of this command and its usage, see Configuring Queue Buffers , on page 108.

Step 10	**queue-limit** {*packets* | **cos** | **dscp** | **percent**}
	Example:
	`Device(config-pmap-c)# queue-limit cos 7 percent 50`
	`Device(config-pmap-c)#`
		(Optional) Specifies the queue maximum threshold for the tail drop:
		• *packets*: Packets by default, enter a value between 1 to 2000000.
		• **cos**: Enter the parameters for each COS value.
		• **dscp**: Enter the parameters for each DSCP value.
		• **percent**: Enter the percentage for the threshold.
		For a more detailed example of this command and its usage, see Configuring Queue Limits, on page 111.

Step 11	**service-policy** *policy-map name*
	Example:
	`Device(config-pmap-c)# service-policy test_2000`
	`Device(config-pmap-c)#`
		(Optional) Configures the QoS service policy.

Step 12	**set** {**cos** | **dscp** | **ip** | **precedence** | **qos-group** | **wlan**}
	Example:
	`Device(config-pmap-c)# set cos 7`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.

	Command or Action	Purpose
		• **ip**: Sets IP specific values. • **precedence**: Sets precedence in IP(v4) and IPv6 packet. • **qos-group**: Sets the QoS Group.

Step 13	**shape average** {*target _bit_rate* | **percent**}
	Example:
	`Device(config-pmap-c) #shape average percent 50`
	`Device(config-pmap-c) #`
		(Optional) Sets the traffic shaping. Command parameters include:
		• *target_bit_rate*: Target bit rate.
		• **percent**: Percentage of interface bandwidth for Committed Information Rate.
		For a more detailed example of this command and its usage, see Configuring Shaping, on page 113.

Step 14	**end**
	Example:
	`Device(config-pmap-c) #end`
	`Device(config-pmap-c) #`
		Saves the configuration changes.

What to do next

Configure the interface.

Configuring Class-Based Packet Marking

This procedure explains how to configure the following class-based packet marking features on your device:

- CoS value
- DSCP value
- IP value
- Precedence value
- QoS group value
- WLAN value

Before you begin

You should have created a class map and a policy map before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy1`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class1`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change.
		Command options for policy class map configuration mode include the following:
		• **bandwidth**: Bandwidth configuration options.
		• **exit**: Exits from the QoS class action configuration mode.
		• **no**: Negates or sets default values for the command.
		• **police**: Policer configuration options.
		• **priority**: Strict scheduling priority configuration options for this class.
		• **queue-buffers**: Queue buffer configuration options.
		• **queue-limit**: Queue maximum threshold for Weighted Tail Drop (WTD) configuration options.
		• **service-policy**: Configures the QoS service policy.
		• **set**: Sets QoS values using the following options:
		  • CoS values
		  • DSCP values
		  • Precedence values
		  • QoS group values

	Command or Action	Purpose
		• WLAN values • **shape**: Traffic-shaping configuration options. **Note** This procedure describes the available configurations using **set** command options. The other command options (**bandwidth**) are described in other sections of this guide. Although this task lists all of the possible **set** commands, only one **set** command is supported per class.

Step 4	**Example:**
	`Device(config-pmap)# set cos 5`
	`Device(config-pmap)#`
		(Optional) Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.
		You can also set the following values using the **set cos** command:
		• **cos table**: Sets the CoS value based on a table map.
		• **dscp table**: Sets the code point value based on a table map.
		• **precedence table**: Sets the code point value based on a table map.
		• **qos-group table**: Sets the CoS value from QoS group based on a table map.

Step 5	**Example:**
	`Device(config-pmap)# set dscp af11`
	`Device(config-pmap)#`
		(Optional) Sets the DSCP value.
		In addition to setting specific DSCP values, you can also set the following using the **set dscp** command:
		• **default**: Matches packets with default DSCP value (000000).
		• **dscp table**: Sets the packet DSCP value from DSCP based on a table map.
		• **ef**: Matches packets with EF DSCP value (101110).
		• **precedence table**: Sets the packet DSCP value from precedence based on a table map.

	Command or Action	Purpose
		• **qos-group table**: Sets the packet DSCP value from a QoS group based upon a table map.

Step 6	**set ip** {**dscp** | **precedence**}
	Example:
	`Device(config-pmap)# set ip dscp c3`
	`Device(config-pmap)#`
		(Optional) Sets IP specific values. These values are either IP DSCP or IP precedence values.
		You can set the following values using the **set ip dscp** command:
		• *dscp value*: Sets a specific DSCP value.
		• **default**: Matches packets with default DSCP value (000000).
		• **dscp table**: Sets the packet DSCP value from DSCP based on a table map.
		• **ef**: Matches packets with EF DSCP value (101110).
		• **precedence table**: Sets the packet DSCP value from precedence based on a table map.
		• **qos-group table**: Sets the packet DSCP value from a QoS group based upon a table map.
		You can set the following values using the **set ip precedence** command:
		• *precedence value*: Sets the precedence value (from 0 to 7) .
		• **cos table**: Sets the packet precedence value from Layer 2 CoS based on a table map.
		• **dscp table**: Sets the packet precedence from DSCP value based on a table map.
		• **precedence table**: Sets the precedence value from precedence based on a table map
		• **qos-group table**: Sets the precedence value from a QoS group based upon a table map.

Step 7	**set precedence** {*precedence value* | **cos table** *table-map name* | **dscp table** *table-map name* | **precedence table** *table-map name* | **qos-group table** *table-map name*}	(Optional) Sets precedence values in IPv4 and IPv6 packets.

	Command or Action	Purpose

	Example:
	`Device(config-pmap)# set precedence 5`
	`Device(config-pmap)#`
		You can set the following values using the **set precedence** command:
		• *precedence value*: Sets the precedence value (from 0 to 7) .
		• **cos table**: Sets the packet precedence value from Layer 2 CoS on a table map.
		• **dscp table**: Sets the packet precedence from DSCP value on a table map.
		• **precedence table**: Sets the precedence value from precedence based on a table map.
		• **qos-group table**: Sets the precedence value from a QoS group based upon a table map.

Step 8	**set qos-group** {*qos-group value* | **dscp table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set qos-group 10`
	`Device(config-pmap)#`
		(Optional) Sets QoS group values. You can set the following values using this command:
		• *qos-group value*: A number from 1 to 31.
		• **dscp table**: Sets the code point value from DSCP based on a table map.
		• **precedence table**: Sets the code point value from precedence based on a table map.

Step 9	**end**
	Example:
	`Device(config-pmap)# end`
	`Device#`
		Saves configuration changes.

Step 10	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Attach the traffic policy to an interface using the **service-policy** command.

Attaching a Traffic Policy to an Interface

After the traffic class and traffic policy are created, you must use the **service-policy** interface configuration command to attach a traffic policy to an interface, and to specify the direction in which the policy should be applied (either on packets coming into the interface or packets leaving the interface).

Before you begin

A traffic class and traffic policy must be created before attaching a traffic policy to an interface.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface** *type*
	Example:
	`Device(config)# interface fortygigabitEthernet1/0/1`
	`Device(config-if)#`
		Enters interface configuration mode and configures an interface.
		Command parameters for the interface configuration include:
		• **TwentyfiveGigabitEthernet**: 25-Gigabit Ethernet
		• **FortyGigabitEthernet**: Forty Gigabit Ethernet
		• **HundredGigabitEthernet**: 100-Gigabit Ethernet
		• **Vlan**: Catalyst VLANs
		Note Tunnel interface is not supported.

Step 3	**service-policy** {**input** *policy-map* | **output** *policy-map*}
	Example:
	`Device(config-if)# service-policy output policy_map_01`
	`Device(config-if)#`
		Attaches a policy map to an input or output interface. This policy map is then used as the service policy for that interface.
		In this example, the traffic policy evaluates all traffic leaving that interface.

Step 4	**end**
	Example:
	`Device(config-if)# end`
	`Device#`
		Saves configuration changes.

	Command or Action	Purpose
Step 5	**show policy map** **Example:** `Device# show policy map`	(Optional) Displays statistics for the policy on the specified interface.

What to do next

Proceed to attach any other traffic policy to an interface, and to specify the direction in which the policy should be applied.

Classifying, Policing, and Marking Traffic on Physical Ports by Using Policy Maps

You can configure a nonhierarchical policy map on a physical port that specifies which traffic class to act on. Actions supported are remarking and policing.

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** {*class-map name* | **match-any** | **match-all**}
	Example:
	`Device(config)# class-map ipclass1`
	`Device(config-cmap)# exit`
	`Device(config)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default.
		• If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.

	Command or Action	Purpose
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** { *access list index* | *access list name* }
	Example:
	`Device(config-cmap)# match access-group 1000`
	`Device(config-cmap)# exit`
	`Device(config)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• group-object
		• ip
		• mpls
		• precedence
		• protocol
		• qos-group
		• vlan
		• wlan
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map ipclass1`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

Step 5	**class** {*class-map-name* | **class-default**}
	Example:
	`Device(config-pmap)# class ipclass1`
	`Device(config-pmap-c)#`
		Defines a traffic classification, and enter policy-map class configuration mode.
		By default, no policy map class-maps are defined.
		If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command.

	Command or Action	Purpose
		A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 6	**set** {**cos** | **dscp** | **ip** | **precedence** | **qos-group** | **wlan user-priority**}
	Example:
	`Device(config-pmap-c)# set dscp 45`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets QoS group.
		In this example, the **set dscp** command classifies the IP traffic by setting a new DSCP value in the packet.

Step 7	**police** {*target_bit_rate* | **cir** | **rate** }
	Example:
	`Device(config-pmap-c)# police 100000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*: Specifies the bit rate per second, enter a value between 8000 and 10000000000.
		• **cir**: Committed Information Rate.
		• **rate**: Specifies the police rate PCR for hierarchical policies.
		In this example, the **police** command adds a policer to the class where any traffic beyond the 100000 set target bit rate is dropped.

Step 8	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 9	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 10	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigabitEthernet 1/0/2`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 11	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input flowit`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map ipclass1`	(Optional) Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

If applicable to your QoS configuration, configure classification, policing, and marking of traffic on SVIs by using policy maps.

Classifying and Marking Traffic by Using Policy Maps

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by using policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**class-map** {*class-map name* | **match-any** | **match-all**} **Example:** `Device(config)# class-map class_vlan100`	Enters class map configuration mode. • Creates a class map to be used for matching packets to the class whose name you specify. • If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. • If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. **Note** This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match vlan** *vlan number*
	Example:
	`Device(config-cmap)# match vlan 100`
	`Device(config-cmap)# exit`
	`Device(config)#`
		Specifies the VLAN to match to the class map.

Step 4	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map policy_vlan100`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

Step 5	**description** *description* **Example:** `Device(config-pmap)# description vlan`	(Optional) Enters a description of the policy map.

	Command or Action	Purpose
	`100`	

Step 6	**class** {*class-map-name* | **class-default**}
	Example:
	`Device(config-pmap)# class class_vlan100`
	`Device(config-pmap-c)#`
		Defines a traffic classification, and enters the policy-map class configuration mode.
		By default, no policy map class-maps are defined.
		If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command.
		A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 7	**set** {**cos** | **dscp** | **ip** | **precedence** | **qos-group** | **wlan user-priority**}
	Example:
	`Device(config-pmap-c)# set dscp af23`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets QoS group.
		In this example, the **set dscp** command classifies the IP traffic by matching the packets with a DSCP value of AF23 (010010).

Step 8	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 9	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 10	**interface** *interface-id* **Example:** `Device(config)# interface hundredgigabitethernet 1/0/3`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 11	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input policy_vlan100`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map`	(Optional) Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Table Maps

Table maps are a form of marking, and also enable the mapping and conversion of one field to another using a table. For example, a table map can be used to map and convert a Layer 2 CoS setting to a precedence value in Layer 3.

Note

- A table map can be referenced in multiple policies or multiple times in the same policy.
- A table map configured for a custom output policy under the default class-map, takes affect for all DSCP traffic regardless of which class map the traffic is classified for. The workaround is to remove the table map and configure the **set dscp** command under the default class to change the DSCP marking for classified traffic. If there is any non-queuing action (policer or marking) on a user-defined class, then the packet retains its value or remarks in the user-defined class itself.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**table-map** *name* {**default** {*default value* | **copy** | **ignore**} | **exit** | **map** {**from** *from value* **to** *to value* } | **no**}
	Example:
	`Device(config)# table-map table01`
	`Device(config-tablemap)#`
		Creates a table map and enters the table map configuration mode. In table map configuration mode, you can perform the following tasks:
		• **default**: Configures the table map default value, or sets the default behavior for a value not found in the table map to copy or ignore.
		• **exit**: Exits from the table map configuration mode.
		• **map**: Maps a *from* to a *to* value in the table map.
		• **no**: Negates or sets the default values of the command.

Step 3	**map from** *value* **to** *value*
	Example:
	`Device(config-tablemap)# map from 0 to 2`
	`Device(config-tablemap)# map from 1 to 4`
	`Device(config-tablemap)# map from 24 to 3`
	`Device(config-tablemap)# map from 40 to 6`
	`Device(config-tablemap)# default 0`
	`Device(config-tablemap)#`
		In this step, packets with DSCP values 0 are marked to the CoS value 2, DSCP value 1 to the CoS value 4, DSCP value 24 to the CoS value 3, DSCP value 40 to the CoS value 6 and all others to the CoS value 0.
		Note The mapping from CoS values to DSCP values in this example is configured by using the **set** policy map class configuration command as described in a later step in this procedure.

Step 4	**exit**
	Example:
	`Device(config-tablemap)# exit`
	`Device(config)#`
		Returns to global configuration mode.

Step 5	**exit** **Example:** `Device(config) exit`	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device#`	

Step 6	**show table-map**
	Example:
	`Device# show table-map`
	`Table Map table01`
	`from 0 to 2`
	`from 1 to 4`
	`from 24 to 3`
	`from 40 to 6`
	`default 0`
		Displays the table map configuration.

Step 7	**configure terminal**
	Example:
	`Device# configure terminal`
	`Device(config)#`
		Enters global configuration mode.

Step 8	**policy-map**
	Example:
	`Device(config)# policy-map table-policy`
	`Device(config-pmap)#`
		Configures the policy map for the table map.

Step 9	**class class-default**
	Example:
	`Device(config-pmap)# class class-default`
	`Device(config-pmap-c)#`
		Matches the class to the system default.

Step 10	**set cos dscp table** *table map name*
	Example:
	`Device(config-pmap-c)# set cos dscp table table01`
	`Device(config-pmap-c)#`
		If this policy is applied on input port, that port will have trust DSCP enabled on that port and marking will take place depending upon the specified table map.

Step 11	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Returns to privileged EXEC mode.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

How to Configure QoS Features and Functionality

The following sections provide configurational information about QoS features and functionality.

Configuring Bandwidth

This procedure explains how to configure bandwidth on your device.

Before you begin

You should have created a class map for bandwidth before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_bandwidth01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_bandwidth01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio* }} **Example:** `Device(config-pmap-c)# bandwidth 200000`	Configures the bandwidth for the policy map. The parameters include: • *Kb/s*: Configures a specific value in kilobits per second (from 100 to 100000000).

	Command or Action	Purpose
	`Device(config-pmap-c)#`	• **percent**: Allocates minimum bandwidth to a particular class based on a percentage. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is equally divided along all bandwidth queues. • **remaining**: Allocates minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than percentages to each queue; the queues will be assigned certain weights which are inline with these ratios. Ratios can range from 0 to 100. Total bandwidth ratio allocation for the policy in this case can exceed 100. **Note** You cannot mix bandwidth types on a policy map. For example, you cannot configure bandwidth in a single policy map using both a bandwidth percent and in kilobits per second.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating the policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Police

This procedure explains how to configure policing on your device.

Before you begin

You should have created a class map for policing before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_police01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_police01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**police** {*target_bit_rate* [*burst bytes* | **bc** | **conform-action** | **pir**] | **cir** {*target_bit_rate* | **percent** *percentage*} | **rate** {*target_bit_rate* | **percent** *percentage*} **conform-action transmit exceed-action** {**drop** [**violate action**] | **set-cos-transmit** | **set-dscp-transmit** | **set-prec-transmit** | **transmit** [**violate action**] }}
	Example:
	`Device(config-pmap-c)# police 8000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		The following **police** subcommand options are available:
		• *target_bit_rate*: Bits per second (from 8000 to 10000000000).
		• *burst bytes*: Enter a value from 1000 to 512000000.
		• **bc**: Conform burst.
		• **conform-action**: Action taken when rate is less than conform burst.
		• **pir**: Peak Information Rate.
		• **cir**: Committed Information Rate.
		• *target_bit_rate*: Target bit rate (8000 to10000000000).

	Command or Action	Purpose
		• **percent**: Percentage of interface bandwidth for CIR. • **rate**: Specifies the police rate, PCR for hierarchical policies, or SCR for single-level ATM 4.0 policer policies. • *target_bit_rate*: Target Bit Rate (8000 to 10000000000). • **percent**: Percentage of interface bandwidth for rate. The following **police conform-action transmit exceed-action** subcommand options are available: • **drop**: Drops the packet. • **set-cos-transmit**: Sets the CoS value and sends it. • **set-dscp-transmit**: Sets the DSCP value and sends it. • **set-prec-transmit**: Rewrites the packet precedence and sends it. • **transmit**: Transmits the packet. **Note** Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies. **Note** The **show policy-map** command output does not display counters for conformed bytes and exceeded bytes

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Priority

This procedure explains how to configure priority on your device.

Note The device supports giving priority to specified queues. There are two priority levels available (1 and 2). Queues supporting voice and video should be assigned a priority level of 1.

Before you begin

You should have created a class map for priority before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_priority01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_priority01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**priority** [*Kb/s* [*burst_in_bytes*] | **level** *level_value* [*Kb/s* [*burst_in_bytes*] | **percent** *percentage* [*burst_in_bytes*]] | **percent** *percentage* [*burst_in_bytes*]]	(Optional) The **priority** command assigns a strict scheduling priority for the class. The command options include:

	Command or Action	Purpose

	Example:
	
	`Device(config-pmap-c)# priority level 1`
	`Device(config-pmap-c)#`
		• *Kb/s*: Specifies the kilobits per second (from 1 to 2000000).
		• *burst_in_bytes*: Specifies the burst in bytes (from 32 to 2000000).
		
		• **level** *level_value*: Specifies the multilevel (1-2) priority queue.
		• *Kb/s*: Specifies the kilobits per second (from 1 to 2000000).
		• *burst_in_bytes*: Specifies the burst in bytes (from 32 to 2000000).
		
		• **percent**: Percentage of the total bandwidth.
		• *burst_in_bytes*: Specifies the burst in bytes (from 32 to 2000000).
		
		• **percent**: Percentage of the total bandwidth.
		• *burst_in_bytes*: Specifies the burst in bytes (32 to 2000000).
		
		Note Priority level 1 is more important than priority level 2. Priority level 1 reserves bandwidth that is processed first for QoS, so its latency is very low. Both priority level 1 and 2 reserve bandwidth.

Step 5	**end**
	Example:
	
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring SGT based QoS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**class-map** *class-map-name* {**match-any** | **match-all** } **Example:** `Device(config)# class-map c1`	Specifies the class-map and enters class-map configuration mode.
Step 3	**match security-group source tag** *sgt-number* **Example:** `Device(config-cmap)# match security-group source tag 1000`	Configures the value for security-group source security tag.
Step 4	**match security-group destination tag** *dgt-number* **Example:** `Device(config-cmap)# match security-group destination tag 2000`	Configures the value for security-group destination security tag.

Step 5	**exit**
	Example:
	`Device(config-cmap)# exit`
	`Device#`
		Exits route-map configuration mode and returns to global configuration mode.

Step 6	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map pin`
	`Device(config-pmap)#`
		Specifies the policy-map and enters policy-map configuration mode.
		policy-map-name is the name of the child policy map. The name can be a maximum of 40 alphanumeric characters.

Step 7	**class** *class-name* **Example:** `Device(config-pmap)# class c1`	Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command

	Command or Action	Purpose
	`Device(config-pmap-c)#`	options for policy class map configuration mode include the following: • *word*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.
Step 8	**set dscp** *dscp-value* **Example:** `Device(config-pmap-c)# set dscp af11`	Configures the Differentiated Services CodePoint (DSCP) value.

Step 9	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes. Exits class-map configuration mode and enters global configuration mode.

Step 10	**interface** *interface-num* **Example:** `Device(config)# interface GigabitEthernet1/0/24`	Specifies the interface and enters the interface configuration mode.
Step 11	**service-policy** { **input** | **output** } *policy-map-name* **Example:** `Device(config-if)#service-policy input pin`	Assigns policy-map to the ingress of the interface.

Step 12	**end**
	Example:
	`Device(config-if)# end`
	`Device#`
		Saves configuration changes. Exits interface configuration mode and enters global configuration mode.

Configuration Example for SGT based QoS Classification

The following is an sample configuration for SGT based QoS on an interface:

```
ip access-list role-based sgt_acl
 10 permit ip
cts role-based sgt-map 24.0.0.0/8 sgt 24
cts role-based enforcement
cts role-based permissions from 24 to 24 sgt_acl

class-map match-all c1
 match protocol attribute business-relevance business-relevant
```

```
 match protocol attribute traffic-class ops-admin-mgmt
 match security-group destination tag 24
 match security-group source tag 24

policy-map pin
 class c1
  set dscp af11
 class class-default
  set dscp af12

interface GigabitEthernet1/0/24
 no switchport
 ip address 24.1.1.2 255.255.255.0
 service-policy input pin
 ip nbar protocol-discovery
```

Configuring Queues and Shaping

The following sections provide configurational information about queueing and shaping.

Configuring Egress Queue Characteristics

Depending on the complexity of your network and your QoS solution, you may need to perform all of the procedures in this section. You need to make decisions about these characteristics:

- Which packets are mapped by DSCP, CoS, or QoS group value to each queue and threshold ID?
- What drop percentage thresholds apply to the queues, and how much reserved and maximum memory is needed for the traffic type?
- How much of the fixed buffer space is allocated to the queues?
- Does the bandwidth of the port need to be rate limited?
- How often should the egress queues be serviced and which technique (shaped, shared, or both) should be used?

Note You can only configure the egress queues on the device.

Configuring Queue Buffers

The device allows you to allocate buffers to queues. If there is no allocation made to buffers, then they are divided equally for all queues. You can use the queue-buffer ratio to divide it in a particular ratio. Since by default DTS (Dynamic Threshold and Scaling) is active on all queues, these are soft buffers.

Note Queue-buffer ratio cannot be configured with a queue-limit.

Before you begin

The following are prerequisites for this procedure:

- You should have created a class map for the queue buffer before beginning this procedure.
- You must have configured either bandwidth, shape, or priority on the policy map prior to configuring the queue buffers.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_queuebuffer01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_queuebuffer01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio value* }}
	Example:
	`Device(config-pmap-c)# bandwidth percent 80`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The command parameters include:
		• *Kb/s*: Use this command to configure a specific value. The range is 20000 to 100000000.
		• **percent**: Allocates a minimum bandwidth to a particular class using a percentage. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the

	Command or Action	Purpose
		bandwidth is equally divided along all bandwidth queues. • **remaining**: Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than a percentage to each queue; the queues will be assigned certain weights that are inline with these ratios. Ratios can range from 0 to 100. Total bandwidth ratio allocation for the policy in this case can exceed 100. **Note** You cannot mix bandwidth types on a policy map.

Step 5	**queue-buffers** {**ratio** *ratio value*}
	Example:
	`Device(config-pmap-c)# queue-buffers ratio 10`
	`Device(config-pmap-c)#`
		Configures the relative buffer size for the queue.
		Note The sum of all configured buffers in a policy must be less than or equal to 100 percent. Unallocated buffers are are evenly distributed to all the remaining queues. Ensure sufficient buffers are allocated to all queues including the priority queues.
		Note Protocol Data Units(PDUs) for network control protocols such as spanning-tree and LACP utilize the priority queue or queue 0 (when a priority queue is not configured). Ensure sufficient buffers are allocated to these queues for the protocols to function.

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

	Command or Action	Purpose
Step 7	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Queue Limits

You use queue limits to configure Weighted Tail Drop (WTD). WTD ensures the configuration of more than one threshold per queue. Each class of service is dropped at a different threshold value to provide for QoS differentiation. With the device, each queue has 3 explicit programmable threshold classes—0, 1, 2. Therefore, the enqueue/drop decision of each packet per queue is determined by the packet's threshold class assignment, which is determined by the DSCP, CoS, or QoS group field of the frame header.

WTD also uses a soft limit, and therefore you are allowed to configure the queue limit to up to 400 percent (maximum four times the reserved buffer from common pool). This soft limit prevents overrunning the common pool without impacting other features.

Note You can only configure queue limits on the device egress queues on wired ports.

Before you begin

The following are prerequisites for this procedure:

- You should have created a class map for the queue limits before beginning this procedure.
- You must have configured either bandwidth, shape, or priority on the policy map prior to configuring the queue limits.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**policy-map** *policy name* **Example:** `Device(config)# policy-map`	Enters policy map configuration mode. Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose
	`policy_queuelimit01` `Device(config-pmap)#`	

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_queuelimit01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio value* }}
	Example:
	`Device(config-pmap-c)# bandwidth 500000`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The parameters include:
		• *Kb/s*: Use this command to configure a specific value. The range is 20000 to 100000000.
		• **percent**: Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is equally divided along all bandwidth queues.
		• **remaining**: Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than a percentage to each queue; the queues will be assigned certain weights that are inline with these ratios. Ratios can range from 0 to 100. Total bandwidth ratio allocation for the policy in this case can exceed 100.
		Note You cannot mix bandwidth types on a policy map.

	Command or Action	Purpose

Step 5	**queue-limit** {*packets* **packets** | **cos** {*cos value* { *maximum threshold value* | **percent** *percentage* } | **values** {*cos value* | **percent** *percentage* } } | **dscp** {*dscp value* {*maximum threshold value* | **percent** *percentage*} | *match packet* {*maximum threshold value* | **percent** *percentage*} | **default** {*maximum threshold value* | **percent** *percentage*} | **ef** {*maximum threshold value* | **percent** *percentage*} | **dscp values** *dscp value*} | **percent** *percentage* }}
	Example:
	`Device(config-pmap-c)# queue-limit dscp 3 percent 20`
	`Device(config-pmap-c)# queue-limit dscp 4 percent 30`
	`Device(config-pmap-c)# queue-limit dscp 5 percent 40`
		Sets the queue limit threshold percentage values.
		With every queue, there are three thresholds (0,1,2), and there are default values for each of these thresholds. Use this command to change the default or any other queue limit threshold setting. For example, if DSCP 3, 4, and 5 packets are being sent into a specific queue in a configuration, then you can use this command to set the threshold percentages for these three DSCP values. For additional information about queue limit threshold values, see #unique_125.
		Note The device does not support absolute queue-limit percentages. The device only supports DSCP or CoS queue-limit percentages.

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 7	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Proceed to configure any additional policy maps for QoS for your network. After creating your policy maps, proceed to attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Shaping

You use the **shape** command to configure shaping (maximum bandwidth) for a particular class. The queue's bandwidth is restricted to this value even though the port has additional bandwidth left. You can configure shaping as an average percent, as well as a shape average value in bits per second.

Before you begin

You should have created a class map for shaping before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_shaping01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_shaping01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**shape average** {*target bit rate* | **percent** *percentage*}
	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Sharped Profile Queuing

This procedure explains how to configure sharped profile queuing on your switch:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy_shaping01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
		policy-map-name is the name of the child policy map. The name can be a maximum of 40 alphanumeric characters.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_shaping01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth** {*Kb/s* | **percent** *percentage* | **remaining** {**ratio** *ratio value*}}
	Example:
	`Device(config-pmap-c)# bandwidth 200000`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map. The parameters include:
		• *Kb/s*: Configures a specific value in kilobits per second (from 100 to 100000000).
		• **percent**: Allocates minimum bandwidth to a particular class based on a percentage. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the

	Command or Action	Purpose
		bandwidth is equally divided along all bandwidth queues. • **remaining**: Allocates minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize entire port bandwidth. The total sum cannot exceed 100 percent. It is preferred to use this command when the **priority** command is used for certain queues in the policy. You can also assign ratios rather than percentages to each queue; the queues will be assigned certain weights which are inline with these ratios. Ratios can range from 1 to 65536. Total bandwidth ratio allocation for the policy in this case can exceed 100. **Note** You cannot mix bandwidth types on a policy map.

Step 5	**shape average** {*target bit rate* | **percent** *percentage*}
	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Sharped Profile Queuing Configuration

The following is the example for sharped queuing:

```
Policy Map test
    Class test1
      bandwidth 20 (%)
      Average Rate Traffic Shaping
      cir 40%
    Class test3
      Average Rate Traffic Shaping
      cir 50%
    Class test2
      Average Rate Traffic Shaping
      cir 50%
```

```
Class test4
  bandwidth 20 (%)
Class test5
  Average Rate Traffic Shaping
  cir 70%
Class test6
  Average Rate Traffic Shaping
  cir 60%
```

Monitoring QoS

The following commands can be used to monitor QoS on the device:

Table 11: Monitoring QoS

Command	Description
show class-map [*class_map_name*]	Displays a list of all class maps configured.
show policy-map [*policy_map_name*]	Displays a list of all policy maps configured. Command parameters include: • policy map name • interface • session
show policy-map interface {**TwentyfiveGigabitEthernet** | **FortyGigabitEthernet** | **HundredGigabitEthernet** |**Vlan**}	Displays the runtime representation and statistics of all the policies configured on the device. Command parameters include: • TwentyfiveGigabitEthernet—25-Gigabit Ethernet • FortyGigabitEthernet—40-Gigabit Ethernet • HundredGigabitEthernet—100-Gigabit Ethernet • Vlan—Catalyst VLANs **Note** Though **wireless** option is visible on the CLI, it is not supported.
show table-map	Displays all the table maps and their configurations.

Configuration Examples for QoS

The following sections provide configuration examples for QoS.

Examples: TCP Protocol Classification

TCP packets can be classified based on port numbers. The configuration for TCP protocol is as follows:

```
Device#show ip acce tcp
Extended IP access list tcp
    10 permit tcp any any eq 80
Device #
Device #show run class-map tcp

Current configuration : 63 bytes
!
class-map match-all tcp
 match access-group name tcp
!
end
Device #
Device #show run policy-map tcp

Current configuration : 56 bytes
!
policy-map tcp
 class tcp
  police 1000000000
!
end
Device #

Device #show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy output tcp
end

Device #
```

Examples: UDP Protocol Classification

UDP packets can be classified based on port numbers. The configuration example for UDP protocol is as follows:

```
Device#show ip acce udp
Extended IP access list udp
    10 permit udp any any eq ntp
Device #

Device #show run class-map udp
Building configuration...

Current configuration : 63 bytes
```

```
!
class-map match-all udp
 match access-group name udp
!
end

Device #
Device #show run policy-map udp
Building configuration...

Current configuration : 56 bytes
!
policy-map udp
 class udp
  police 1000000000
!
end
Device #
Device #show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy output udp
end

Device #
```

Examples: RTP Protocol Classification

RTP packets can be classified based on port numbers. The configuration example for RTP protocol is as follows:

```
Device# show ip access-list rtp
Extended IP access list rtp
    10 permit udp any any eq 554
    11 permit tcp any any eq 554
Device #

Device #show run class-map rtp

Current configuration : 63 bytes
!
class-map match-all rtp
 match access-group name rtp
!
end

Device #
Device #show run policy-map rtp

Current configuration : 56 bytes
!
policy-map rtp
 class rtp
  police 1000000000
!
end

Device #
Device #show run int tw 1/0/1
```

```
Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy output rtp
end

Device #
```

Examples: Classification by Access Control Lists

This example shows how to classify packets for QoS by using access control lists (ACLs):

```
Device# configure terminal
Device(config)# access-list 101 permit ip host 12.4.1.1 host 15.2.1.1
Device(config)# class-map acl-101
Device(config-cmap)# description match on access-list 101
Device(config-cmap)# match access-group 101
Device(config-cmap)#
```

After creating a class map by using an ACL, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service Layer 2 Classification

This example shows how to classify packets for QoS using a class of service Layer 2 classification:

```
Device# configure terminal
Device(config)# class-map cos
Device(config-cmap)# match cos ?
  <0-7>  Enter up to 4 class-of-service values separated by white-spaces
Device(config-cmap)# match cos 3 4 5
Device(config-cmap)#
```

After creating a class map by using a CoS Layer 2 classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service DSCP Classification

This example shows how to classify packets for QoS using a class of service DSCP classification:

```
Device# configure terminal
Device(config)# class-map dscp
Device(config-cmap)# match dscp af21 af22 af23
Device(config-cmap)#
```

After creating a class map by using a DSCP classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: VLAN ID Layer 2 Classification

This example shows how to classify for QoS using a VLAN ID Layer 2 classification:

```
Device# configure terminal
Device(config)# class-map vlan-120
Device(config-cmap)# match vlan ?
  <1-4095>  VLAN id
Device(config-cmap)# match vlan 120
Device(config-cmap)#
```

After creating a class map by using a VLAN Layer 2 classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Classification by DSCP or Precedence Values

This example shows how to classify packets by using DSCP or precedence values:

```
Device# configure terminal
Device(config)# class-map prec2
Device(config-cmap)# description matching precedence 2 packets
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit
Device(config)# class-map ef
Device(config-cmap)# description EF traffic
Device(config-cmap)# match ip dscp ef
Device(config-cmap)#
```

After creating a class map by using a DSCP or precedence values, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Hierarchical Policy Configuration

The following is an example of a configuration using hierarchical polices:

```
Device# configure terminal
Device(config)# class-map c1
Device(config-cmap)# match dscp 30
Device(config-cmap)# exit

Device(config)# class-map c2
Device(config-cmap)# match precedence 4
Device(config-cmap)# exit

Device(config)# class-map c3
Device(config-cmap)# exit

Device(config)# policy-map child
Device(config-pmap)# class c1
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# police rate percent 20 conform-action transmit exceed action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit

Device(config-pmap)# class c2
```

```
Device(config-pmap-c)# bandwidth 20000
Device(config-pmap-c)# exit
Device(config-pmap)# class class-default
Device(config-pmap-c)# bandwidth 20000
Device(config-pmap-c)# exit
Device(config-pmap)# exit

Device(config)# policy-map parent
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 1000000
Device(config-pmap-c)# service-policy child
Device(config-pmap-c)# end
```

The following example shows a hierarchical policy using table maps:

```
Device(config)# table-map dscp2dscp
Device(config-tablemap)# default copy
Device(config)# policy-map ssid_child_policy
Device(config-pmap)# class voice
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# police 15000000
Device(config-pmap)# class video
Device(config-pmap-c)# priority level 2
Device(config-pmap-c)# police 10000000
Device(config)# policy-map ssid_policy
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 30000000
Device(config-pmap-c)# queue-buffer ratio 0
Device(config-pmap-c)# set dscp dscp table dscp2dscp
Device(config-pmap-c)# service-policy ssid_child_policy
```

Examples: Classification for Voice and Video

This example describes how to classify packet streams for voice and video using device specific information.

In this example, voice and video are coming in from end-point A into HundredGigabitEthernet1/0/1on the device and have precedence values of 5 and 6, respectively. Additionally, voice and video are also coming from end-point B into FortyGigabitEthernet1/0/2 on the device with DSCP values of EF and AF11, respectively.

Assume that all the packets from the both the interfaces are sent on the uplink interface, and there is a requirement to police voice to 100 Mbps and video to 150 Mbps.

To classify per the above requirements, a class to match voice packets coming in on HundredGigabitEthernet1/0/1 is created, named voice-interface-1, which matches precedence 5. Similarly another class for voice is created, named voice-interface-2, which will match voice packets in HundredGigabitEthernet1/0/3. These classes are associated to two separate policies named input-interface-1, which is attached to HundredGigabitEthernet1/0/1, and input-interface-2, which is attached toHundredGigabitEthernet1/0/3. The action for this class is to mark the qos-group to 10. To match packets with QoS-group 10 on the output interface, a class named voice is created which matches on QoS-group 10. This is then associated to another policy named output-interface, which is associated to the uplink interface. Video is handled in the same way, but matches on QoS-group 20.

The following example shows how classify using the above device specific information:

```
Device(config)#
Device(config)# class-map voice-interface-1
Device(config-cmap)# match ip precedence 5
```

```
Device(config-cmap)# exit

Device(config)# class-map video-interface-1
Device(config-cmap)# match ip precedence 6
Device(config-cmap)# exit

Device(config)# class-map voice-interface-2
Device(config-cmap)# match ip dscp ef
Device(config-cmap)# exit

Device(config)# class-map video-interface-2
Device(config-cmap)# match ip dscp af11
Device(config-cmap)# exit

Device(config)# policy-map input-interface-1
Device(config-pmap)# class voice-interface-1
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# exit

Device(config-pmap)# class video-interface-1
Device(config-pmap-c)# set qos-group 20

Device(config-pmap-c)# policy-map input-interface-2
Device(config-pmap)# class voice-interface-2
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# class video-interface-2
Device(config-pmap-c)# set qos-group 20
Device(config-pmap-c)# exit
Device(config-pmap)# exit

Device(config)# class-map voice
Device(config-cmap)# match qos-group 10
Device(config-cmap)# exit

Device(config)# class-map video
Device(config-cmap)# match qos-group 20
Device(config)# policy-map output-interface
Device(config-pmap)# class voice
Device(config-pmap-c)# police 256000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit

Device(config-pmap)# class video
Device(config-pmap-c)# police 1024000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit
```

Examples: Average Rate Shaping Configuration

The following example shows how to configure average rate shaping:

```
Device# configure terminal
Device(config)# class-map prec1
Device(config-cmap)# description matching precedence 1 packets
Device(config-cmap)# match ip precedence 1
Device(config-cmap)# end

Device# configure terminal
Device(config)# class-map prec2
```

```
Device(config-cmap)# description matching precedence 2 packets
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit

Device(config)# policy-map shaper
Device(config-pmap)# class prec1
Device(config-pmap-c)# shape average 512000
Device(config-pmap-c)# exit

Device(config-pmap)# policy-map shaper
Device(config-pmap)# class prec2
Device(config-pmap-c)# shape average 512000
Device(config-pmap-c)# exit

Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 1024000
```

After configuring the class maps, policy map, and shape averages for your configuration, proceed to then apply the policy map to the interface for QoS.

Examples: Queue-limit Configuration

The following example shows how to configure a queue-limit policy based upon DSCP values and percentages:

```
Device# configure terminal
Device#(config)# policy-map port-queue
Device#(config-pmap)# class dscp-1-2-3
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 1 percent 80
Device#(config-pmap-c)# queue-limit dscp 2 percent 90
Device#(config-pmap-c)# queue-limit dscp 3 percent 100
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-4-5-6
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 4 percent 20
Device#(config-pmap-c)# queue-limit dscp 5 percent 30
Device#(config-pmap-c)# queue-limit dscp 6 percent 20
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-7-8-9
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 7 percent 20
Device#(config-pmap-c)# queue-limit dscp 8 percent 30
Device#(config-pmap-c)# queue-limit dscp 9 percent 20
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-10-11-12
Device#(config-pmap-c)# bandwidth percent 20
Device#(config-pmap-c)# queue-limit dscp 10 percent 20
Device#(config-pmap-c)# queue-limit dscp 11 percent 30
Device#(config-pmap-c)# queue-limit dscp 12 percent 20
Device#(config-pmap-c)# exit

Device#(config-pmap)# class dscp-13-14-15
Device#(config-pmap-c)# bandwidth percent 10
Device#(config-pmap-c)# queue-limit dscp 13 percent 20
Device#(config-pmap-c)# queue-limit dscp 14 percent 30
Device#(config-pmap-c)# queue-limit dscp 15 percent 20
```

```
Device#(config-pmap-c)# end
Device#
```

After finishing with the above policy map queue-limit configuration, you can then proceed to apply the policy map to an interface for QoS.

Examples: Queue Buffers Configuration

The following example shows how configure a queue buffer policy and then apply it to an interface for QoS:

```
Device# configure terminal
Device(config)# policy-map policy1001
Device(config-pmap)# class class1001
Device(config-pmap-c)# bandwidth remaining ratio 10
Device(config-pmap-c)# queue-buffer ratio ?
  <0-100>  Queue-buffers ratio limit
Device(config-pmap-c)# queue-buffer ratio 20
Device(config-pmap-c)# end

Device# configure terminal
Device(config)# interface HundredGigabitE1/0/3
Device(config-if)# service-policy output policy1001
Device(config-if)# end
```

Examples: Policing Action Configuration

The following example displays the various policing actions that can be associated to the policer. These actions are accomplished using the conforming, exceeding, or violating packet configurations. You have the flexibility to drop, mark and transmit, or transmit packets that have exceeded or violated a traffic profile.

For example, a common deployment scenario is one where the enterprise customer polices traffic exiting the network towards the service provider and marks the conforming, exceeding and violating packets with different DSCP values. The service provider could then choose to drop the packets marked with the exceeded and violated DSCP values under cases of congestion, but may choose to transmit them when bandwidth is available.

Note The Layer 2 fields can be marked to include the CoS fields, and the Layer 3 fields can be marked to include the precedence and the DSCP fields.

One useful feature is the ability to associate multiple actions with an event. For example, you could set the precedence bit and the CoS for all conforming packets. A submode for an action configuration could then be provided by the policing feature.

This is an example of a policing action configuration:

```
Device# configure terminal
Device(config)# policy-map police
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 1000000 pir 2000000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
```

```
Device(config-pmap-c-police)# end
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Examples: Policer VLAN Configuration

The following example displays a VLAN policer configuration. At the end of this configuration, the VLAN policy map is applied to an interface for QoS.

```
Device# configure terminal
Device(config)# class-map vlan100
Device(config-cmap)# match vlan 100
Device(config-cmap)# exit
Device(config)# policy-map vlan100
Device(config-pmap)# policy-map class vlan100
Device(config-pmap-c)# police 100000 bc conform-action transmit exceed-action drop
Device(config-pmap-c-police)# end
Device# configure terminal
Device(config)# interface HundredGigabitE1/0/5
Device(config-if)#  service-policy input vlan100
```

Examples: Policing Units

The policing unit is the basis on which the token bucket works. CIR and PIR are specified in bits per second. The burst parameters are specified in bytes. This is the default mode; it is the unit that is assumed when no units are specified. The CIR and PIR can also be configured in percent, in which case the burst parameters have to be configured in milliseconds.

The following is an example of a policer configuration in bits per second. In this configuration, a dual-rate three-color policer is configured where the units of measurement is bits. The burst and peak burst are all specified in bits.

```
Device(config)# policy-map bps-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police rate 100000 peak-rate 1000000
conform-action transmit exceed-action set-dscp-transmit dscp table
DSCP_EXCE violate-action drop
```

Examples: Single-Rate Two-Color Policing Configuration

The following example shows how to configure a single-rate two-color policer:

```
Device(config)# class-map match-any prec1
Device(config-cmap)# match ip precedence 1
Device(config-cmap)# exit
```

```
Device(config)# policy-map policer
Device(config-pmap)# class prec1
Device(config-pmap-c)# police cir 256000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

Examples: Dual-Rate Three-Color Policing Configuration

The following example shows how to configure a dual-rate three-color policer:

```
Device# configure terminal
Device(config)# policy-Map dual-rate-3color-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 64000 bc 2000 pir 128000 be 2000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Examples: Table Map Marking Configuration

The following steps and examples show how to use table map marking for your QoS configuration:

1. Define the table map.

   Define the table-map using the **table-map** command and indicate the mapping of the values. This table does not know of the policies or classes within which it will be used. The default command in the table map indicates the value to be copied into the 'to' field when there is no matching 'from' field. In the example, a table map named table-map1 is created. The mapping defined is to convert the value from 0 to 1 and from 2 to 3, while setting the default value to 4.

   ```
   Device(config)# table-map table-map1
   Device(config-tablemap)# map from 0 to 1
   Device(config-tablemap)# map from 2 to 3
   Device(config-tablemap)# default 4
   Device(config-tablemap)# exit
   ```

2. Define the policy map where the table map will be used.

   In the example, the incoming CoS is mapped to the DSCP based on the mapping specified in the table table-map1. For this example, if the incoming packet has a DSCP of 0, the CoS in the packet is set 1. If no table map name is specified the command assumes a default behavior where the value is copied as is from the 'from' field (DSCP in this case) to the 'to' field (CoS in this case). Note however, that while the

CoS is a 3-bit field, the DSCP is a 6-bit field, which implies that the CoS is copied to the first three bits in the DSCP.

```
Device(config)# policy map policy1
Device(config-pmap)# class class-default
Device(config-pmap-c)# set cos dscp table table-map1
Device(config-pmap-c)# exit
```

3. Associate the policy to an interface.

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# service-policy output policy1
Device(config-if)# exit
```

Example: Table Map Configuration to Retain CoS Markings

The following example shows how to use table maps to retain CoS markings on an interface for your QoS configuration.

The cos-trust-policy policy (configured in the example) is enabled in the ingress direction to retain the CoS marking coming into the interface. If the policy is not enabled, only the DSCP is trusted by default. If a pure Layer 2 packet arrives at the interface, then the CoS value will be rewritten to 0 when there is no such policy in the ingress port for CoS.

```
Device# configure terminal
Device(config)# table-map cos2cos
Device(config-tablemap)# default copy
Device(config-tablemap)# exit


Device(config)# policy map cos-trust-policy
Device(config-pmap)# class class-default
Device(config-pmap-c)# set cos cos table cos2cos
Device(config-pmap-c)# exit

Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# service-policy input cos-trust-policy
Device(config-if)# exit
```

Where to Go Next

Review the auto-QoS documentation to see if you can use these automated capabilities for your QoS configuration.

Additional References for QoS

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	*Command Reference (* *Cisco IOS Quality of*

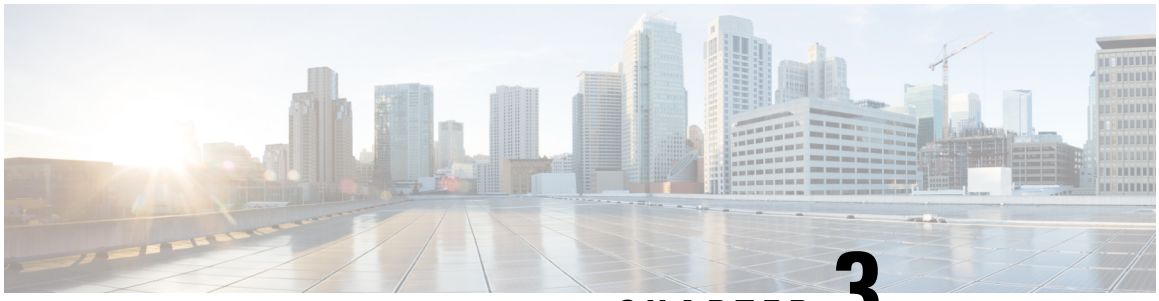
Feature History for QoS

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	QoS Functionality	QoS provides preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size. **Note** This release does not support converged access.
Cisco IOS XE Amsterdam 17.2.1	Unified Buffer Sharing	Buffer sharing between cores was introduced.
Cisco IOS XE Cupertino 17.7.1	Low priority control packet mapping to Non-Low Latency Queueing (LLQ)	The system generated low-priority CPU traffic is now mapped to threshold 2 of a non-priority queue with highest bandwidth.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to Cisco Feature Navigator.



CHAPTER 3

Configuring QoS on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2)

Prerequisites for QoS

Before configuring standard Quality of Service (QoS), you must have a thorough understanding of these items:

- Standard QoS concepts.
- Classic Cisco IOS QoS.
- Understanding of QoS implementation.
- Understanding of Virtual output queueing (V0Q) architecture.
- Types of applications used and the traffic patterns on your network.
- Traffic characteristics and needs of your network. For example, is the traffic on your network bursty? Do you need to reserve bandwidth for voice and video streams?
- Bandwidth requirements and speed of the network.
- Location of congestion points in the network.

Restrictions for QoS on Wired Targets

A target is an entity where a policy is applied, and a wired target can be a port.

The following are the restrictions for applying QoS features on a device for a wired target:

- For a type queueing policymap, a maximum of eight queuing classes are supported on a device port for a wired target.
- For a nonqueueing policymap, policer is only supported in ingress. Up to 32 policers are supported per ingress policymap per port. Each policer has counter for conform, exceed, and violate.
- A maximum of only 1599 policymaps can be created.
- A total of eight priority levels (0 to 7) are supported, but only one priority level can be configured for each classmap. A class without priority level configured is priority level 0, which is referred to as nonpriority class.
- In a hierarchical policy, overlapping actions between parent and child are not allowed, except when a policy has the port shaper in the parent policy and the queuing features in the child policy.
- For hierarchical QoS (H-QoS) applied in the ingress direction, policing in both the parent and the child is not supported in a QoS hierarchy.
- Marking in both the parent and the child is not supported in a QoS hierarchy.
- Empty classes are supported.
- The conform action must be transmit under a policer within a policy map.
- Marking action is not supported in the egress type queueing policy. To support remark on egress, a new policy map must be attached, which can be based on DSCP or PREC or CoS or MPLS EXP or QoS-group, but not based on ACL.
- For Generic Routing Encapsulation (GRE) tunnel interface, a policymap can be attached only to physical members, and not on the logical tunnel interface.
- Classification counters have the following specific restrictions:
  - Classification counters count packets instead of bytes.
  - Filter-based classification counters are not supported.
  - Only QoS configurations with marking or policing can trigger the classification counter.
  - The classification counter is only port based, and aggregation is not performed.
  - As long as there is policing or marking action in the policy, the class will have classification counters.
  - When there are multiple match statements in a class, the traffic counter is cumulative for all the match statements in the class.
  - A class without policer in ingress or egress policymap is assigned to a classification counter. For an ingress policy, 32 unique counters per port are supported, and for egress policy, eight unique counters per port are supported. If an ingress or egress policy has more than 32/8 classes without policers, the extra classes will share the same match counter. A maximum of 256 classes are supported per policy on the wired port for the wired target.

- The device supports 15 unique combinations of policer exceed markdown and policer violate markdown tables. A policymap must use the same combination, which means different classes in the same policymap must use the same table map for exceed markdown. The same is applicable for violate markdown.
- Overlapping and marking actions are not supported in H-QoS policy.
- Policer value can only be configured in either the parent or the child, not both.
- Queuing actions are supported only on DSCP, CoS, QoS-group, IP precedence, and EXP based classification.
- Application Visibility and Control (AVC) and Network Based Application Recognition (NBAR) based QoS are not supported.
- Classification is not supported for the following:
  - Virtual Private LAN Services (VPLS)
  - Layer 2 and MAC
  - Packet length for fixed and range
  - Real-Time Transport Protocol (RTP) for header and type
  - Access control entries (ACEs)
- Conditional markdown using multiple table-maps in the same policy-map is not supported.
- GRE tunnel QoS for policing and marking is not supported.
- Locator ID Separation Protocol (LISP) QoS is not supported.
- QoS ACL ternary content addressable memory (TCAM) for egress is not supported.
- QoS metadata for App-ID entries are not supported.
- Security group tag (SGT) aware QoS is not supported.
- StackWise Virtual Link (SVL) QOS is not supported.
- Policing in egress direction is not supported.
- Policing and queueing are not supported on SVI and tunnel interfaces.
- Object group-based ACLs with QoS ACL based classification are not supported.
- Remark with ACL based classification is not supported in egress direction.
- For Broadcast, Unknown unicast and Multicast (BUM) traffic, QoS Queue statistics visibility is not supported in **show policy-map type queue interface** command.
- Only 2 output queuing statistics can be viewed for multicast traffic.
- Only 8 queuing per port are supported for queuing of traffic.
- Queuing policy-map can be classified only using traffic-class, and the set option in queuing policy-map is not supported.
- Classification using IPv6 based ACE using tcp flag in it is not supported

- Egress remarking or set option is supported based on DSCP, CoS, precedence, MPLS EXP, or QoS-group. A new policy-map must be defined and applied on the interface in egress direction. In case of MPLS, the egress remarking policy on label imposition node works only if there is an EXP based marking policy on the ingress interface.
- The table map should be of the same QoS type tag. For example, table map from DSCP to CoS and CoS to precedence are not supported.
- Aggregate policing for egress is not supported.
- Policing is not supported for outgoing packets.
- Control-plane policy packet counter statistics update is not supported.
- For hierarchical QoS (HQoS) police applied in the ingress direction, child-level policing is not supported. Policing of traffic is based on the police rate value defined in parent-level policing.
- Egress police with QoS group classification along with DSCP/PREC/COS/EXP based classification are not supported.

The following are the restrictions and considerations for applying QoS features on EtherChannel and channel member interfaces:

- Queuing policy is supported only on EtherChannel member ports, and nonqueuing policy is supported only on EtherChannel interface.
- When a nonqueuing service policy is applied to channel members, it gets rejected with a message:

```
Only queueing type policy is supported on member interfaces.
```

- Auto QoS is not supported on EtherChannel members.

Restrictions for Egress Queuing Policy

- Each class-map matches only one of the traffic classes, the range of which is 0 to 7. **class-default** always matches traffic class 0. Also, no other class can match traffic class 0.
- If there is no **set traffic-class** in the ingress policy map, the QoS tag value is mapped to traffic class 0 or 7 by default.
- To select queues that are different from the system default mapping, configure the traffic class using the ingress policy map. The traffic class that is configured affects VOQ selection for all the egress ports.
- Each unique combination of traffic-classes in a type queuing policy-map requires a separate traffic class profile. The number of traffic class profiles are limited to eight for main interfaces.
- On Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) statistics for multicast and Layer 2 traffic dropped by queuing police, due to exceeded shape limit value, are not displayed in the total packet drop statistics on the interface. It displays statistics only for unicast traffic.
- Each priority level, when configured, must be configured to the class that matches the corresponding traffic class as shown in the following table:

Note This is applicable only to priority level-based queuing policy map.

Table 12: Priority Level to Traffic Class Mapping

Priority Level	Traffic Class
P1 (highest)	7
P2	6
P3	5
P4	4
P5	3
P6	2
P7	1
None (lowest)	0

- Each priority level—ranging from 1 to 7—can be configured to one class only. That is, there cannot be more than one class at the same priority level. Each class that does not have a priority level configured, is referred to as priority normal. Note that there can be multiple priority normals.
- Priority level must start from 1. Therefore, if there is only one priority class, it must be priority level 1.
- If all the priority levels that are configured in a policy map are sorted, they must be contiguous. In other words, you cannot skip a priority level. For example, P1, P2, P4 (skipping P3), is not allowed. This condition can be met after adding or removing class from a type queueing policy map.
- On Cisco Catalyst 9500X Series Switches, you can not attach a class-map that matches both, DSCP and MPLS experimental bit (EXP), to an ingress policy map.
- Only these actions are supported in the queuing policy:
  - Priority
  - Shape

Note The set option is not available in the queueing policy map.

  - Bandwidth remaining ratio
  - Queue-limit
  - Random Early Detection (RED)
  - Match

Information About QoS

The following sections provide information about QoS.

QoS Components

QoS consists of the following key components:

- Classification: This is the process of distinguishing one type of traffic from another based upon access control lists (ACLs), Differentiated Services Code Point (DSCP), Class of Service (CoS), and other factors.
- Marking and mutation: Marking is used on traffic to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another. When traffic is marked, QoS operations on that traffic can be applied. This can be accomplished directly using the **set** command or through a table map, which takes input values and translates them directly to values in the output.
- Shaping: This is the process of imposing a maximum rate of traffic while regulating the traffic rate in such a way that downstream devices are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface.
- Policing: This is used to impose a maximum rate on a traffic class. If the rate is exceeded, then a specific action is taken as soon as the event occurs.

Note Policing of traffic is supported only in the ingress direction.

- Queuing: This is used to prevent traffic congestion. Traffic is sent to specific queues for servicing and scheduling, based upon bandwidth allocation. Traffic is then scheduled or sent out through the port. Traffic class is used as the classification value for queueing.

QoS Terminology

The following terms are used interchangeably in this QoS configuration guide:

- Upstream (direction towards the device) is the same as ingress.
- Downstream (direction from the device) is the same as egress.

Information About QoS

A default policy is always present for the device, and the traffic-class and discard-class are set based on the default policy. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size.

The following are specific features provided by QoS:

- Low latency
- Bandwidth guarantee

- Buffering capabilities and dropping disciplines
- Traffic policing
- Enables the changing of the attribute of the frame or packet header
- Relative services

Modular QoS CLI

QoS features are enabled through the Modular QoS CLI (MQC). The MQC is a CLI structure that allows you to create traffic policies and attach these policies to interfaces. A traffic policy contains a traffic class and one or more QoS features. A traffic class is used to classify traffic, while the QoS features in the traffic policy determine how to treat the classified traffic. One of the main goals of MQC is to provide a platform-independent interface for configuring QoS across Cisco platforms.

Virtual Output Queuing

In traditional methods of traffic management, traffic is sent to the egress output queues without taking into consideration the egress interface availability to transmit. Therein lies the problem as well. In case of traffic congestion, traffic may get dropped at the egress port. That means the network resources spent getting the packets from the ingress input queue across the switch fabric to the output queues at egress are wasted. That is not all—the same input queue buffers traffic meant for different egress ports, so congestion on one egress port could affect traffic on another port, an event referred to as head-of-line-blocking. Virtual output queueing (VOQ) resolves this problem.

In a VOQ architecture, a separate virtual queue for each egress port is maintained for the physical buffer of each input port. That is, unlike in the traditional method where one input queue will direct traffic to multiple output queues, every output queue will have a dedicated virtual queue. Therefore, in case of congestion, virtual queue only for that particular egress port will be blocked and transmitted only when the egress port has enough resources.

Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) usesVOQ architecture in accordance with the Cisco IOS Modular QoS CLI (MQC).

Supported QoS Features for Wired Access

The following table describes the supported QoS features for wired access.

Table 13: Supported QoS Features for Wired Access

Feature	Description
Supported targets	• 10-Gigabit Ethernet • 40-Gigabit Ethernet • 25-Gigabit Ethernet • 100-Gigabit Ethernet • EtherChannel member ports **Note** Queuing is supported only on EtherChannel member ports and policing is supported on port-channel interface. • SVI **Note** It is supported only in Layer 2 member ports on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP- 2).
Configuration sequence	QoS policy installed using the **service-policy** command.
Supported number of queues at port level	Up to eight queues supported on a port.

Supported classification mechanism	• DSCP
	• IP precedence
	• MPLS experimental bits (EXP)
	• CoS
	• QoS group
	• ACL membership including:
	  • IPv4 ACLs
	  • IPv6 ACLS

Hierarchical QoS

H-QoS allows you to perform:

- Hierarchical classification: Traffic classification is based upon traffic class.

- Hierarchical shaping: Shaping can also be configured at multiple levels in the hierarchy.

Note For the parent you only have a configuration class default, and the only action for the class default, is shaping.

QoS Implementation

Typically, networks operate on a best-effort delivery basis, which means that all traffic has equal priority and an equal chance of being delivered in a timely manner. When congestion occurs, all traffic has an equal chance of being dropped.

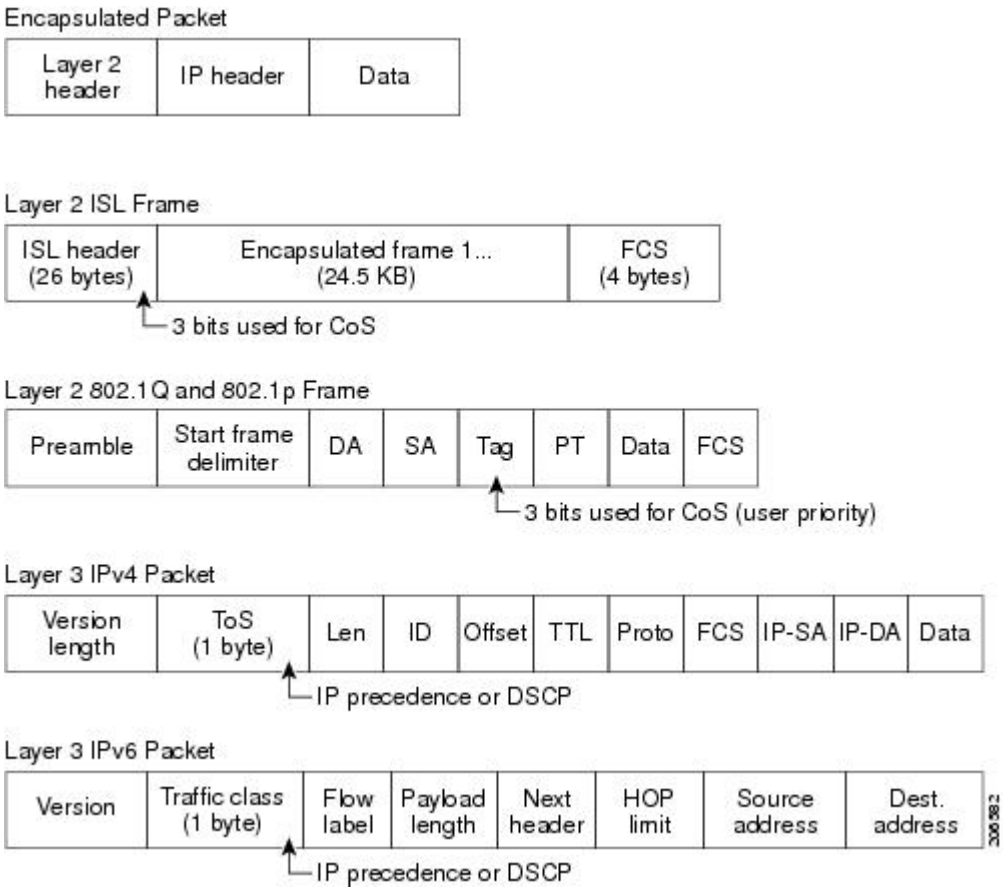
When you configure the QoS feature, you can select specific network traffic, prioritize it according to its relative importance, and use congestion-management and congestion-avoidance techniques to provide preferential treatment. Implementing QoS in your network makes network performance more predictable and bandwidth utilization more effective.

QoS implementation is based on the Differentiated Services (Diff-Serv) architecture, a standard from the Internet Engineering Task Force (IETF). This architecture specifies that each packet is classified upon entry into the network.

The classification is carried in the IP packet header, using six bits from the deprecated IP type of service (ToS) field to carry the classification (*class*) information. Classification can also be carried in the Layer 2 frame.

The special bits in the Layer 2 frame or a Layer 3 packet are shown in the following figure:

Figure 4: Classification Layers of QoS in Frames and Packets



Layer 2 Frame Prioritization Bits

Layer 2 Inter-Switch Link (ISL) frame headers have a 1-byte User field that carries an IEEE 802.1p class of service (CoS) value in the three least-significant bits. On ports configured as Layer 2 ISL trunks, all the traffic is in ISL frames.

Layer 2 802.1Q frame headers have a 2-byte Tag Control Information field that carries the CoS value in the three most-significant bits, which are called User Priority bits. On ports configured as Layer 2 802.1Q trunks, all the traffic is in 802.1Q frames, except for the traffic in the native VLAN.

Other frame types cannot carry Layer 2 CoS values.

Layer 2 CoS values range from 0 for low priority to 7 for high priority.

Layer 3 Packet Prioritization Bits

Layer 3 IP packets can carry either an IP precedence value or a Differentiated Services Code Point (DSCP) value. QoS supports the use of either value because DSCP values are backward-compatible with IP precedence values.

IP precedence values range from 0 to 7. DSCP values range from 0 to 63.

End-to-End QoS Solution Using Classification

All the switches and the routers that access the internet rely on class information to provide the same forwarding treatment to packets with the same class information, and different treatment to packets with different class information. The class information in the packet can be assigned by end hosts or by switches or routers along the way, based on a configured policy, detailed examination of the packet, or both. Detailed examination of the packet is expected to occur closer to the edge of the network, so that the core switches and routers are not overloaded with this task.

Switches and routers along the path can use the class information to limit the amount of resources allocated per traffic class. The behavior of an individual device when handling traffic in the Diff-Serv architecture is called per-hop behavior. If all the devices along a path provide a consistent per-hop behavior, you can construct an end-to-end QoS solution.

Implementing QoS in your network can be a simple task or a complex task, depending on the QoS features offered by your internetworking devices, the traffic types and patterns in your network, and the granularity of control that you need over incoming and outgoing traffic.

Packet Classification

Packet classification is the process of identifying a packet as belonging to one of several classes in a defined policy, based on certain criteria. The Modular QoS CLI (MQC) is a policy-class based language. The policy class language is used to define the following:

- Class map template with one or several match criteria
- Policy map template with one or several classes associated to the policy map

The policy map template is then associated to one or several interfaces on the device.

Packet classification is the process of identifying a packet as belonging to one of the classes defined in the policy map. The process of classification will exit when the packet being processed matches a specific filter in a class. This is referred to as first-match exit. If a packet matches multiple classes in a policy, irrespective of the order of classes in the policy map, it will still exit the classification process after matching the first class.

If a packet does not match any of the classes in the policy, it is classified into the default class in the policy. Every policy map has a default class, which is a system-defined class to match the packets that do not match any of the user-defined classes.

Packet classification can be categorized into the following types:

- Classification based on information that is propagated with the packet
- Classification based on information that is device specific
- Hierarchical classification

Classification Based on Information Propagated with a Packet

Classification based on information that is part of a packet, and propagated either end-to-end or between hops, typically includes the following:

- Classification based on Layer 3 or 4 headers
- Classification based on Layer 2 information

Classification Based on Layer 3 or Layer 4 Header

This is the most common deployment scenario. Numerous fields in the Layer 3 and Layer 4 headers can be used for packet classification.

At the most granular level, this classification methodology can be used to match an entire flow. For this deployment type, an access control list (ACL) can be used. ACLs can also be used based on various subsets of the flow, for example, source IP address only, destination IP address only, or a combination of both.

Classification can also be done based on the precedence or DSCP values in the IP header. The IP precedence field is used to indicate the relative priority with which a particular packet needs to be handled. It is made up of three bits in the IP header's type of service (ToS) byte.

The following table shows the different IP precedence bit values and their names.

Table 14: IP Precedence Values and Names

IP Precedence Value	IP Precedence Bits	IP Precedence Names
0	000	Routine
1	001	Priority
2	010	Immediate
3	011	Flash
4	100	Flash Override
5	101	Critical
6	110	Internetwork control
7	111	Network control

Note All the routing control traffic in a network use the IP precedence value 6 by default. IP precedence value 7 is also reserved for network control traffic. Therefore, we do not recommend the use of IP precedence values 6 and 7 for user traffic.

The DSCP field is made up of 6 bits in the IP header, and is being standardized by the Internet Engineering Task Force (IETF) Differentiated Services Working Group. The original ToS byte, which contained the DSCP bits, has been renamed the DSCP byte. The DSCP field is part of the IP header, similar to IP precedence. The DSCP field is a super set of the IP precedence field. Therefore, the DSCP field is used and is set in ways that are similar to what was described with respect to IP precedence.

Note

- The DSCP field definition is backward-compatible with the IP precedence values.
- Some fields in the Layer 2 header can also be set using a policy.

Classification Based on Layer 2 Header

The CoS method can be used to perform classification based on the Layer 2 header information. Classification is based on the three bits in the Layer 2 header based on the IEEE 802.1p standard. This usually maps to the ToS byte in the IP header.

Classification Based on Information that is Device Specific

A device also provides classification mechanisms in scenarios that are available where classification is not based on information in the packet header or payload.

At times you might be required to aggregate traffic coming from multiple input interfaces into a specific class in the output interface. For example, multiple customer edge routers might be going into the same access device on different interfaces. The service provider might want to police all the aggregate voice traffic going into the core to a specific rate. However, the voice traffic coming in from the different customers could have different ToS settings. QoS group-based classification is a feature that is useful in these scenarios.

Policies configured on the input interfaces set the QoS group to a specific value, which can then be used to classify the packets in the policies enabled on the output interface.

The QoS group is a field in the packet data structure that is internal to a device. It is important to note that a QoS group is an internal label to the device and is not a part of the packet header.

QoS Wired Model

To implement QoS, the device must perform the following tasks:

- Traffic classification: Distinguish packets or flows from one another.
- Traffic marking and policing: Assign a label to indicate the given quality of service as the packets move through the device, and then make the packets comply with the configured resource usage limits.
- Queuing and scheduling: Provide different treatment in all the situations where resource contention exists.
- Shaping: Ensure that the traffic sent from the device meets a specific traffic profile.

Ingress Port Activity

The following activities occur at the ingress port of a device:

- Classification: Classifying a distinct path for a packet by associating it with a QoS label. For example, the device maps the CoS or DSCP in the packet to a QoS label to distinguish one type of traffic from another. The QoS label that is generated identifies all future QoS actions to be performed on this packet.
- Marking: Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and determines what to do with the packet (pass through a packet without modification, mark down the QoS label in the packet, or drop the packet). Traffic can be marked with traffic-class to hit a particular VoQ. The traffic-class ranges from 0 to 7. Marking can also be done without policing.

Egress Port Activity

The following activities occur at the egress port of the device:

- Policing: Policing is not supported for outgoing packets.

- Marking: Marking rewrites the QoS tags such as DSCP and CoS in the packet so that the next hop can apply QoS based on the new value. You must configure a separate policy-map (different from the type queueing policy-map described in the next bullet point) and attach it to the egress direction of an interface by using the **service-policy output policy-name** command. This policy-map can only contain marking action using the **set qos-tag value** command. For example, `set dscp cs6` or `set cos 5`.
- Queuing: Queuing selects which VoQ to use, and applies congestion management such as queue-limit and random-detect. Packets admitted into VoQ are scheduled to the corresponding egress output queueing. Shape, priority, and bandwidth remaining ratio affects how packets are scheduled. VoQ selection happens on the ingress side by setting the traffic-class in the ingress policy if configured, or it follows the default QoS tag to traffic-class mapping which only uses traffic-class 0 and 7. On the egress side, a separate **type queueing** policy-map can be attached to the interface unless the default queuing configuration is acceptable. This policy-map can only use class-maps that match traffic-class (class-default matches traffic-class 0 by default). The **service-policy type queueing output policy-name** command is used to attach the queuing policy-map to an interface. The policy-name used must be a policy-name defined using the **policy-map type queueing** command.

Note Queuing is only supported using traffic class.

Classification

Classification is the process of distinguishing one kind of traffic from another by examining the fields in the packet. Classification is enabled only if QoS is enabled on a device. By default, QoS is enabled in a device.

During classification, a device performs a lookup and assigns a QoS label to a packet. The QoS label identifies all the QoS actions to be performed on a packet and from which queue the packet is sent.

Access Control Lists

You can use IP standard and extended IP to define a group of packets with the same characteristics (class). You can also classify IP traffic based on IPv6 ACLs.

In the QoS context, the permit and deny actions in the access control entries (ACEs) have different meanings from security ACLs:

- If a match with a permit action is encountered (first-match principle), the specified QoS-related action is taken.
- If multiple ACLs are configured on a port, the lookup stops after the packet matches the first ACL with a permit action, and QoS processing begins.

Note Deny-based ACL classification is not supported. Only permit-based ACL is supported.

After a class map has been defined with the ACL, you can use the class map name in a policy map. A policy might contain multiple classes, with actions specified for each one of them. A policy might include commands to classify the class as a particular aggregate, for example, assign a DSCP, or rate-limit the class. This policy is then attached to a particular port on which it becomes effective.

Note Layer 2-based ACLs are not supported.

Class Maps

A class map is a mechanism that you use to name a specific traffic flow (or class) and isolate it from all other traffic. The class map defines the criteria used to match against a specific traffic flow to further classify it. The criteria can include matching the access group defined by the ACL or matching a specific list of DSCP or IP precedence values or CoS values or traffic-class values. If you have more than one type of traffic that you want to classify, you can create another class map and use a different name.

Note You can create multiple ACLs in one class. By default the class criteria is **match-all**. You cannot use ACLs with one matching TCP and UDP for **match-all** as one packet cannot be TCP and UDP at the same time. You can use it with **match-any** which can match all TCP and UDP packets.

You create a class map by using the **class-map** global configuration command or the **class** policy-map configuration command. You should use the **class-map** command when the map is shared among mulitple policies. When you enter the **class-map** command, the device enters the class-map configuration mode.

You can create a default class by using the **class class-default** policy-map configuration command. The default class is system-defined and cannot be configured. Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as default traffic.

Note When configuring a class-map for traffic-class, only one match traffic-class is supported.

Time-to-Live Classification

Note IPv6 time-to-live (TTL) is not supported.

You can classify packets based on the ACL map. You can set TTL as a criterion in the ACL list and perform a TTL check on the incoming packet. The access control entry is used to check the IPv4 TTL to match the value on the incoming packet. The classified packet is either marked or policed based on the policy-map action. Queueing cannot be configured on this classification.

The following is an example of TTL classification:

```
policy-map TTL_MATCH
 class IPV4_TTL
  police rate 6000000000
  set dscp af23

ip access-list extended IPV4_TTL
 permit ip any any ttl eq 100
 permit tcp any any ttl ne 150

!
Device#show run class-map IPV4_TTL
```

```
class-map match-all IPV4_TTL
 match access-group name IPV4_TTL
!

Device#show policy-map interface hun1/0/47

HundredGigE1/0/47

  Service-policy output: TTL_MATCH

    Class-map: IPV4_TTL (match-all)
    553567424 packets
    Match: access-group name IPV4_TTL
    police:
    rate 6000000000 bps, burst 187500000 bytes
    conformed 22983406600 bytes; actions:
    transmit
    exceeded 32375773000 bytes; actions:
    drop
    conformed 588922000 bps, exceeded 830894000 bps
    QoS Set
    dscp af23

    Class-map: class-default (match-any)
    2184433710 packets
    Match: any
```

Policy Maps

A policy map specifies which traffic class to act on. Actions can include the following:

- Setting a specific DSCP or IP precedence value in the class-map
- Setting a CoS value in the class-map
- Setting a traffic-class in the class-map
- Specifying the traffic bandwidth limitations and the action to take when the traffic is out of profile

Before a policy map can be effective, you must attach it to a port.

You create and name a policy map using the **policy-map** global configuration command. When you enter this command, the device enters the policy-map configuration mode. In this mode, you specify the actions to take on a specific traffic class by using the **class** or **set** policy-map configuration and policy-map class configuration commands.

The policy map can also be configured using the **police** and **bandwidth** policy-map class configuration commands, which define the policer, the bandwidth limitations of the traffic, and the action to take if the limits are exceeded. In addition, the policy-map can further be configured using the **priority** policy-map class configuration command, to schedule priority for the class or the queuing policy-map class configuration command, **queue-limit**.

To enable the policy map, you attach it to a port by using the **service-policy** interface configuration command.

Priority based queuing is supported and only in egress direction. Policy-map can be applied to an interface in ingress direction only.

The following is an example of how to configure policy-map which can be applied to an interface in ingress direction only.

```
Device# configure terminal
Device(config)# class-map match-all cs2
```

```
Device(config-cmap)# match dscp cs2
Device(config-cmap)# exit
Device(config)# class-map match-all cs3
Device(config-cmap)# match dscp cs3
Device(config-cmap)# exit
Device(config)# policy-map mul4
Device(config-pmap)# class cs2
Device(config-pmap-c)# police 1000000000
Device(config-pmap-c)# set dscp cs7
Device(config-pmap-c)# exit
Device(config-pmap)# class cs3
Device(config-pmap-c)# set traffic-class 7
Device(config-pmap-c)# set dscp cs5
Device(config-pmap-c)# police cir 40000000000
Device(config-pmap-c)# conform-action transmit
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/2
Device(config-if)# service-policy input pmap
```

Policy Map on Physical Port

A policy map also has these characteristics:

- A policy map can contain multiple class statements, each with different match criteria and policers.
- A policy map can contain a predefined default traffic class explicitly placed at the end of the map.

  When you configure a default traffic class by using the **class class-default** policy-map configuration command, unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as the default traffic class (**class-default**).
- A separate policy-map class can exist for each type of traffic received through a port.

Policing

After a packet is classified and has a DSCP-based, CoS-based, or QoS-group label assigned to it, the policing and marking process can begin.

Policing involves creating a policer that specifies the bandwidth limits for the traffic. Packets that exceed the limits are *out of profile* or *nonconforming*. Each policer decides on a packet-by-packet basis whether the packet is in or out of profile and specifies the actions on the packet. These actions, carried out by the marker, include passing through the packet without modification, dropping the packet, or modifying (marking down) the assigned DSCP or CoS value of the packet.

To avoid out-of-order packets, both conform and nonconforming traffic typically exit the same queue.

Note All traffic, regardless of whether it is bridged or routed, is subjected to a policer, if one is configured. As a result, packets might be dropped or might have their DSCP or CoS fields modified when they are policed and marked.

After you configure the policy map and policing actions, attach the policy-map to a port in ingress direction by using the **service-policy** interface configuration command.

Token-Bucket Algorithm

Policing uses a token-bucket algorithm. As each frame is received by the device, a token is added to the bucket. The bucket has a hole in it and leaks at a rate that you specify as the average traffic rate in bits per second. Each time a token is added to the bucket, the device verifies that there is enough room in the bucket. If there is not enough room, the packet is marked as nonconforming, and the specified policer action is taken (dropped or marked down).

How quickly the bucket fills is a function of the bucket depth (burst-byte), the rate at which the tokens are removed (rate-bps), and the duration of the burst above the average rate. The size of the bucket imposes an upper limit on the burst length and limits the number of frames that can be transmitted back-to-back. If the burst is short, the bucket does not overflow, and no action is taken against the traffic flow. However, if a burst is long and at a higher rate, the bucket overflows, and the policing actions are taken against the frames in that burst.

Marking

Marking is used to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another.

Marking can be used to set certain field/bits in the packet headers, or marking can also be used to set certain fields in the packet structure that is internal to the device. It can also be used with traffic classes for packets to hit specific VoQs. Additionally, the marking feature can be used to define mapping between fields. The following marking methods are available for QoS:

- Packet header
- Device specific information
- Table maps

Packet Header Marking

Marking on fields in the packet header can be classified into two general categories:

- IPv4/v6 header bit marking
- Layer 2 header bit marking

Note Only CoS-based is supported. VLAN ID based is not supported.

The marking feature at the IP level is used to set the precedence or the DSCP in the IP header to a specific value to get a specific per-hop behavior at the downstream device (switch or router), or it can also be used to aggregate traffic from different input interfaces into a single class in the output interface. The functionality is currently supported on both the IPv4 and IPv6 headers.

Marking in the Layer 2 headers is typically used to influence dropping behavior in the downstream devices (switch or router). It works in tandem with the match on the Layer 2 headers. The bits in the Layer 2 header that can be set using a policy map are class of service.

Note Based on whether the packet is Layer 2 or Layer 3, DSCP or CoS bit is marked. On a routed traffic, only DSCP marking is allowed, and on a bridged traffic, only CoS marking is allowed.

Switch-Specific Information Marking

This form of marking includes marking of fields in the packet data structure that are not part of the packets header, so that the marking can be used later in the data path. This is not propagated between the switches. Marking of QoS group falls into this category. This form of marking is only supported in policies that are enabled on the input interfaces. The corresponding matching mechanism can be enabled on the output interfaces on the same switch and an appropriate QoS action can be applied.

Traffic-class can be set in the ingress policy to direct the matched traffic into a corresponding VoQ. Discard-class can be set in the ingress policy to be used later for VoQ congestion management for selecting the corresponding queue-limit or random-detect settings based on discard-class.

Table Map Marking

Table map marking enables the mapping and conversion from one field to another using a conversion table. This conversion table is called a table map.

Depending upon the table map attached to an interface, CoS, DSCP, and Precedence values of the packet are rewritten. The device allows configuring both ingress table map policies and egress table map policies. Table maps can also be used to map or set the incoming traffic to a particular VoQ using traffic-class values (0 to 7). CoS or DSCP or PREC to discard-class on the ingress and QoS group to CoS or DSCP or PREC on the egress are also supported.

Only table-map values of same type of QoS tags are supported. For example, table-map of type CoS to CoS or DSCP to DSCP or PREC to PREC is supported, and also table-map of type DSCP or CoS or PREC to QoS group and DSCP or CoS or PREC to traffic-class is supported.

A table map-based policy supports the following capabilities:

- Mutation: You can have a table map that maps from one DSCP value set to another DSCP value set, and this can be attached to noth ingress and egress ports.
- Rewrite: Packets coming in are rewritten depending upon the configured table map.
- Mapping: Table map based policies can be used instead of set policies.

The following steps are required for table map marking:

1. Define the table map: Use the **table-map** global configuration command to map the values. The table does not know of the policies or classes within which it will be used. The default command in the table map is used to indicate the value to be copied into the to field when there is no matching from field.
2. Define the policy map: You must define the policy map where the table map will be used.
3. Associate the policy to an interface.

Note

- For bridged traffic the from type should be CoS and for routed traffic the from type should be DSCP or PREC. Both can be configured for SVI member interfaces. CoS or DSCP will be selected automatically in the data plane based on how traffic is forwarded.
- DSCP to COS table-maps are not supported.
- When you map a QoS group to a DSCP table used for DSCP marking at egress, the QoS group does not add the equivalent of CoS in the packet. Configure a separate QoS group to COS table map if you want to define the QoS group to a non-zero COS value.

Traffic Conditioning

To support QoS in a network, traffic entering the service provider network needs to be policed on the network boundary routers to ensure that the traffic rate stays within the service limit. Even if a few routers at the network boundary start sending more traffic than what the network core is provisioned to handle, the increased traffic load leads to network congestion. The degraded performance in the network makes it difficult to deliver QoS for all the network traffic.

Traffic policing functions (using the police feature) and shaping functions (using the traffic shaping feature) manage the traffic rate, but differ in how they treat traffic when tokens are exhausted. The concept of tokens comes from the token bucket scheme, a traffic metering function.

Note When running QoS tests on network traffic, you may see different results for the shaper and policing data. Network traffic data from shaping provides more accurate results.

This table compares the policing and shaping functions.

Table 15: Comparison Between Policing and Shaping Functions

Policing Function	Shaping Function
Sends conforming traffic up to the configured rate and allows bursts.	Smooths traffic and sends it out at a constant rate.
When tokens are exhausted, action is taken immediately.	When tokens are exhausted, it buffers packets and sends them out later, when tokens are available. A class with shaping has a queue associated with it which will be used to buffer the packets.
Policing has multiple units of configuration – in bits per second and packets per second. **Note** Policing in packets per second (PPS) is only supported for traffic punted to the CPU, and not for network facing interfaces.	Shaping has only one unit of configuration - in bits per second.

Policing Function	Shaping Function
Policing has multiple possible actions associated with an event, marking and dropping being example of such actions.	Shaping does not have the provision to mark packets that do not meet the profile. A new policy map must be available, which can then be applied on the egress interface.
Works for input traffic only.	Implemented for output traffic only.
Transmission Control Protocol (TCP) detects the line at line speed but adapts to the configured rate when a packet drop occurs by lowering its window size.	TCP can detect that it has a lower speed line and adapt its retransmission timer accordingly. This results in less scope of retransmissions and is TCP-friendly.

Policing

The QoS policing feature is used to impose a maximum rate on a traffic class. If the rate is exceeded, then a specific action is taken as soon as the event occurs. The rate parameters (committed information rate [CIR] and peak information rate [PIR]) are configured in bits per second, and the burst parameters (conformed burst size [B_c] and extended burst size [B_e]) are configured in bytes per second.

Only the single-rate two-color policing forms or policers are supported for QoS.

Single-Rate Two-Color Policing

Single-rate two-color policer is the mode in which you configure only a CIR.

The B_c is an optional parameter, and if it is not specified it is computed by default. In this mode, when an incoming packet has enough tokens available, the packet is considered to be conforming.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 148.

Dual-Rate Three-Color Policing

With the dual rate policer, the device supports only color-blind mode. In this mode, you configure a committed information rate (CIR) and a peak information rate (PIR). As the name suggests, there are two token buckets in this case, one for the peak rate, and one for the conformed rate. The device also supports color-aware mode which can be set using **set discard class 1** action on the ingress classification.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 148.

In the color-blind mode, the incoming packet is first checked against the peak rate bucket. If there are not enough tokens available, the packet is said to violate the rate. If there are enough tokens available, then the tokens in the conformed rate buckets are checked to determine if there are enough tokens available. The tokens in the peak rate bucket are decremented by the size of the packet. If it does not have enough tokens available, the packet is said to have exceeded the configured rate. If there are enough tokens available, then the packet is said to conform, and the tokens in both the buckets are decremented by the size of the packet.

The rate at which tokens are replenished depends on the packet arrival. Assume that a packet comes in at time T1 and the next one comes in at time T2. The time interval between T1 and T2 determines the number of tokens that need to be added to the token bucket. This is calculated as:

Time interval between packets (T2-T1) * CIR)/8 bytes

Shaping

Shaping is the process of imposing a maximum rate of traffic, while regulating the traffic rate in such a way that the downstream switches and routers are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface.

Shaping has a buffer associated with it that ensures that packets which do not have enough tokens are buffered as opposed to being immediately dropped. The number of buffers available to the subset of traffic being shaped is limited and is computed based on a variety of factors. The number of buffers available can also be tuned using specific QoS commands. Packets are buffered as buffers are available, beyond which they are dropped.

Class-Based Traffic Shaping

The device uses class-based traffic shaping. This shaping feature is enabled on a class in a policy that is associated to an interface. A class that has shaping configured is allocated a number of buffers to hold the packets that do not have tokens. The buffered packets are sent out from the class using FIFO. In the most common form of usage, class-based shaping is used to impose a maximum rate for an physical interface or logical interface as a whole. The following shaping forms are supported in a class:

- Average rate shaping
- Hierarchical shaping

Shaping is implemented using a token bucket. The values of CIR, B_c and B_e determine the rate at which the packets are sent out and the rate at which the tokens are replenished.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 65.

Average Rate Shaping

You use the **shape average** policy-map class command to configure average rate shaping.

This command configures a maximum bandwidth for a particular class. The queue bandwidth is restricted to this value even though the port has more bandwidth available. The device supports configuring shape average by either a percentage or by a target bit rate value.

Hierarchical Shaping

Shaping can also be configured at multiple levels in a hierarchy. This is accomplished by creating a parent policy with shaping configured, and then attaching child policies with additional shaping configurations to the parent policy.

The supported hierarchical shaping type is Port Shaper.

The port shaper uses the class-default and the only action permitted in the parent is shaping. The queuing action is in the child with the port shaper.

Queuing and Scheduling

The device uses both queuing and scheduling to help prevent traffic congestion. The device supports the following queuing and scheduling features:

- Bandwidth
- Weighted Tail Drop
- Priority queues
- Queue buffers
- Weighted Random Early Detection

When you define a queuing policy on a port, control packets are mapped to the best priority queue with the highest threshold, which is traffic-class 7. Control packets queue mapping works differently in the following scenarios:

- Without a quality of service (QoS) policy: If no QoS policy is configured, control packets with DSCP values 16, 24, 48, and 56 are mapped to traffic-class 7, which is the highest threshold.
- With an user-defined policy: An user-defined queuing policy configured on egress ports can affect the default priority queue setting on control packets.

  By default each QoS tag with DSPC or CoS or PREC values are matched to a specific traffic-class.

Note Queuing policy in egress direction does not support **match access-group** classification. It also does not support QoS tag based classification such as match DSCP and COS. Class-maps used in queuing policy can only match traffic-class.

Control traffic is redirected to the best queue based on the following rules:

1. The queue corresponding to traffic-class 7 is always the highest priority queue even without **priority level 1** configured.
2. Without ingress policy or if ingress policy does not set traffic-class, DSCP 16, 24, 32, 40, 46, 48, 56 are mapped to traffic-class 7. CoS 4, 5 are mapped to traffic-class 7. The other DSCP and CoS values are mapped to traffic-class 0.
3. You can apply ingress policy and set traffic-class explicitly in order to override the above default mapping. You should make sure that voice and control plane traffic gets mapped to high priority traffic-class.
4. If traffic-class is not configured, the best queue is selected based on the default implementation logic. Cisco IOS software assigns control packets with Differentiated Services Code Point (DSCP) values 16, 24, 48, and 56 are mapped to traffic-class 7 and reassigns the rest of the control traffic in the best queue to traffic-class 0.

Note To provide proper QoS for Layer 3 packets, you must ensure that packets are explicitly classified into appropriate queues.

On a network interface with queueing police enabled, when both unicast and multicast traffic are directed to the same traffic class queue, traffic prioritization happens such that 80% of the available bandwidth is allocated for unicast traffic, while the remaining 20% is reserved for multicast traffic. This allocation creates a traffic distribution ratio of 80:20 between unicast and multicast traffic.

Bandwidth

The device supports the bandwidth remaining ratio configuration. You can use the **bandwidth remaining ratio** policy-map class command to configure the ratio for sharing excess bandwidth. The range is from 1 to 63.

Priority Queues

Each port supports eight ingress queues, of which seven can be given a priority.

You can use the **priority level** policy class-map command to configure the priority for seven classes, including the class-default which is the lowest priority level. Each priority level ranging from 1 to 7 can be configured to one class only, that is, there cannot be more than one class at the same priority level. Each class that does not have a priority level configured is referred to as priority normal, and there can be multiple priority normal classes.

If a priority level is configured in a policy-map, priority level 1 must be configured. For example, policy-map with priority level 2 and priority level 3 but no priority level 1 is not allowed. If all the priority levels configured in a policy-map is sorted, they must be contiguous, that is, no priority level should be skipped, for example, the sequence of priority levels 1, 2, 4, and 5 where priority level 3 is skipped is not allowed.

Each priority level must be configured to the class that matches the corresponding traffic class as shown in the following table:

Table 16: Priority Level to Traffic Class Mapping

Priority Level	Traffic Class
1 (highest)	7
2	6
3	5
4	4
5	3
6	2
7	1
None (lowest)	0

Weighted Random Early Detection

Weighted random early detection (WRED) is a mechanism to avoid congestion in networks. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion, thus avoiding large number of packet drops at once.

For more information about WRED, see Configuring Weighted Random Early Detection, on page 227.

Default Wired QoS Configuration

There are two queues configured by default on each wired interface on the device. All control traffic traverses and is processed through VoQ 7. All other traffic traverses and is processed through VoQ 0.

DSCP Maps

This section provides information about DSCP maps.

Default IP-Precedence-to-DSCP Map

You use the IP-precedence-to-DSCP map to map IP precedence values in incoming packets to a DSCP value that QoS uses internally to represent the priority of the traffic. The following table shows the default IP-precedence-to-DSCP map. If these values are not appropriate for your network, you need to modify them.

Table 17: Default IP-Precedence-to-DSCP Map

IP Precedence Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Default QoS Mapping

The following tables list default QoS mapping values. These values are applicable only if there are no policy-map applied to an interface.

Table 18: DSCP Default Mapping Table for Ingress

DSCP	DSCP	Encapsulation	QoS-group	Table Map	Drop Precedence
0	0	0	0	0	G
1	1	0	0	0	G
2	2	0	0	0	G
3	3	0	0	0	G
4	4	0	0	0	G
5	5	0	0	0	G

DSCP	DSCP	Encapsulation	QoS-group	Table Map	Drop Rate
6	6	0	0	0	G
7	7	0	0	0	G
8	8	1	0	0	G
9	9	1	0	0	G
10	10	1	0	0	G
11	11	1	0	0	G
12	12	1	0	0	G
13	13	1	0	0	G
14	14	1	0	0	G
15	15	1	0	0	G
16	16	2	0	7	G
17	17	2	0	0	G
18	18	2	0	0	G
19	19	2	0	0	G
20	20	2	0	0	G
21	21	2	0	0	G
22	22	2	0	0	G
23	23	2	0	0	G
24	24	3	0	7	G
25	25	3	0	0	G
26	26	3	0	0	G
27	27	3	0	0	G
28	28	3	0	0	G
29	29	3	0	0	G
30	30	3	0	0	G
31	31	3	0	0	G
32	32	4	0	7	G
33	33	4	0	0	G
34	34	4	0	0	G
35	35	4	0	0	G
36	36	4	0	0	G
37	37	4	0	0	G

DSCP	DSCP	Encapsulation	QoS-group	Table Map	Drop Behavior
38	38	4	0	0	G
39	39	4	0	0	G
40	40	5	0	7	G
41	41	5	0	0	G
42	42	5	0	0	G
43	43	5	0	0	G
44	44	5	0	0	G
45	45	5	0	0	G
46	46	5	0	7	G
47	47	5	0	0	G
48	48	6	0	7	G
49	49	6	0	0	G
50	50	6	0	0	G
51	51	6	0	0	G
52	52	6	0	0	G
53	53	6	0	0	G
54	54	6	0	0	G
55	55	6	0	0	G
56	56	7	0	7	G
57	57	7	0	0	G
58	58	7	0	0	G
59	59	7	0	0	G
60	60	7	0	0	G
61	61	7	0	0	G
62	62	7	0	0	G
63	63	7	0	0	G

Table 19: CoS-Drop Eligible Indicator Mapping Table for Ingress

CoS-DEI	CoS-DEI	Encapsulation	QoS-Group	Table Map	Drop Behavior
0	0	0	0	0	G
1	1	0	0	0	G

CoS-DEI	CoS-DEI	Encapsulation	QoS-Group	Table Map	Drop Precedence
2	2	1	0	0	G
3	3	1	0	0	G
4	4	2	0	0	G
5	5	2	0	0	G
6	6	3	0	0	G
7	7	3	0	0	G
8	8	4	0	7	G
9	9	4	0	7	G
10	10	5	0	7	G
11	11	5	0	7	G
12	12	6	0	0	G
13	13	6	0	0	G
14	14	7	0	0	G
15	15	7	0	0	G

Table 20: MPLS-Experimental Mapping Table for Ingress

MPLS Experimental	MPLS Experimental	Encapsulation	QoS-Group	Table Map	Drop Precedence
0	0	0	0	0	G
1	1	1	0	0	G
2	2	2	0	0	G
3	3	3	0	0	G
4	4	4	0	0	G
5	5	5	0	0	G
6	6	6	0	0	G
7	7	7	0	0	G

Table 21: DSCP Mapping Table for Egress

DSCP	DSCP	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
0	0	0	0	0
1	1	4	0	0

DSCP	DSCP	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
2	2	8	0	0
3	3	12	0	0
4	4	16	0	0
5	5	20	0	0
6	6	24	0	0
7	7	28	0	0
8	8	32	1	1
9	9	36	1	1
10	10	40	1	1
11	11	44	1	1
12	12	48	1	1
13	13	52	1	1
14	14	56	1	1
15	15	60	1	1
16	16	64	2	2
17	17	68	2	2
18	18	72	2	2
19	19	76	2	2
20	20	80	2	2
21	21	84	2	2
22	22	88	2	2
23	23	92	2	2
24	24	96	3	3
25	25	100	3	3
26	26	104	3	3
27	27	108	3	3
28	28	112	3	3
29	29	116	3	3
30	30	120	3	3
31	31	124	3	3
32	32	128	4	4

DSCP	DSCP	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
33	33	132	4	4
34	34	136	4	4
35	35	140	4	4
36	36	144	4	4
37	37	148	4	4
38	38	152	4	4
39	39	156	4	4
40	40	160	5	5
41	41	164	5	5
42	42	168	5	5
43	43	172	5	5
44	44	176	5	5
45	45	180	5	5
46	46	184	5	5
47	47	188	5	5
48	48	192	6	6
49	49	196	6	6
50	50	200	6	6
51	51	204	6	6
52	52	208	6	6
53	53	212	6	6
54	54	216	6	6
55	55	220	6	6
56	56	224	7	7
57	57	228	7	7
58	58	232	7	7
59	59	236	7	7
60	60	240	7	7
61	61	244	7	7
62	62	248	7	7
63	63	252	7	7

Table 22: CoS-DEI Mapping Table for Egress

CoS-DEI	CoS-DEI	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
0	0	0	0	0
1	1	0	0	0
2	2	32	1	1
3	3	32	1	1
4	4	64	2	2
5	5	64	2	2
6	6	96	3	3
7	7	96	3	3
8	8	128	4	4
9	9	128	4	4
10	10	160	5	5
11	11	160	5	5
12	12	192	6	6
13	13	192	6	6
14	14	224	7	7
15	15	224	7	7

Table 23: MPLS-Experimental Mapping Table for Egress

MPLS-Experimental	MPLS-Experimental	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
0	0	0	0	0
1	0	32	1	1
2	0	64	2	2
3	0	96	3	3
4	0	128	4	4
5	0	160	5	5
6	0	192	6	6
7	0	224	7	7

How to Configure QoS

How to Configure Class, Policy, and Maps

The following sections provide configuration information about class, policy, and maps.

Creating a Traffic Class

To create a traffic class containing match criteria, use the **class-map** command to specify the traffic class name, and then use the following **match** commands in class-map configuration mode, as needed.

Before you begin

All match commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.

Note Individual class-maps must be defined for different kind of QoS tag values. Matching a traffic-class with two different traffic-class values in the same class-map is not supported. Also, matching traffic-class with other QoS tag values is not supported. For example, matching DSCP or PRE or CoS cannot be added to a class-map which already has a traffic-class match.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** *class-map name* {**match-any** | **match-all**}
	Example:
	`Device(config)# class-map test_1000`
	`Device(config-cmap)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• **match-any:** Any one of the match criteria must be met for traffic entering the traffic class to be classified as part of it.
		• **match-all:** All of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.

	Command or Action	Purpose
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** {*index number* | *name*}
	Example:
	`Device(config-cmap)# match access-group 100`
	`Device(config-cmap)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• ip
		• mpls
		• precedence
		• qos-group
		• traffic-class
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**match cos** *cos value*
	Example:
	`Device(config-cmap)# match cos 2 3 4 5`
	`Device(config-cmap)#`
		(Optional) Matches IEEE 802.1Q or ISL class of service (user) priority values.
		• Enters up to 4 CoS values separated by spaces (0 to 7).

Step 5	**match dscp** *dscp value*
	Example:
	`Device(config-cmap)# match dscp af11 af12`
	`Device(config-cmap)#`
		(Optional) Matches the DSCP values in IPv4 and IPv6 packets.

Step 6	**match ip** {**dscp** *dscp value* | **precedence** *precedence value* } **Example:** `Device(config-cmap)# match ip dscp af11`	(Optional) Matches IP values including the following: • **dscp**: Matches IP DSCP (DiffServ codepoints).

	Command or Action	Purpose
	`af12` `Device(config-cmap)#`	• **precedence**: Matches IP precedence (0 to 7). **Note** Since CPU generated packets are not marked at egress, the packet will not match the configured class-map.

Step 7	**match mpls experimental topmost** *experimental value*
	Example:
	`Device(config-cmap)# match mpls experimental topmost 3`
	`Device(config-cmap)#`
		(Optional) Match MPLS experimental value on topmost label (from 0 to 7).

Step 8	**match qos-group** *qos group value*
	Example:
	`Device(config-cmap)# match qos-group 10`
	`Device(config-cmap)#`
		(Optional) Matches QoS group value (from 0 to 31).

Step 9	**match traffic-class** *traffic class value*
	Example:
	`Device(config-cmap)# match traffic-class 7`
	`Device(config-cmap)#`
		(Optional) Matches QoS traffic class value (from 1 to 7).

Step 10	**end** **Example:** `Device(config-cmap)# end`	Saves the configuration changes.

What to do next

Configure the policy map.

Creating a Traffic Policy

To create a traffic policy, use the **policy-map** global configuration command to specify the traffic policy name.

The traffic class is associated with the traffic policy when the **class** command is used. The **class** command must be entered after you enter the policy map configuration mode. After entering the **class** command, the

device is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

The following policy map class-actions are supported:

- exit: Exits from the QoS class action configuration mode.
- no: Negates or sets default values for the command.
- police: Policer configuration options.
- service-policy: Configures the QoS service policy.
- set: Sets QoS values using the following options:
  - CoS values
  - Discard-class values
  - DSCP values
  - IP values
  - MPLS exp values
  - Precedence values
  - QoS group values
  - Traffic class values

Note If the **set** option is not used in the egress policy to remark egress traffic, the statistics of the policy will not be displayed when you use the **show policy-map type queue interface** and **show policy-map interface** commands.

Before you begin

You should have first created a class map.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**policy-map** *policy-map name* **Example:** `Device(config)# policy-map test_2000`	Enters policy map configuration mode. Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose
	`Device(config-pmap)#`	

Step 3	**class** {*class-name* | **class-default**}
	Example:
	`Device(config-pmap)# class test_1000`
	`Device(config-pmap-c)#`
		Specifies the name of the class whose policy you want to create or change.
		You can also create a system default class for unclassified packets.

Step 4	**police** {*target_bit_rate* | **cir** | **rate**}
	Example:
	`Device(config-pmap-c)# police 100000`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*: Enter the bit rate per second, enter a value between 8000 and 400000000000.
		• **cir**: Committed Information Rate.
		• **rate**: Specify police rate, PCR for hierarchical policies or SCR for single-level ATM 4.0 policer policies.

Step 5	**service-policy** *policy-map name*
	Example:
	`Device(config-pmap-c)# service-policy test_2000`
	`Device(config-pmap-c)#`
		(Optional) Configures the QoS service policy.

Step 6	**set** {**cos** | **discard-class** | **dscp** | **ip** | **mpls** | **precedence** | **qos-group** | **traffic-class**}
	Example:
	`Device(config-pmap-c)# set cos 7`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **discard-class**: Sets discard behavior identifier.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **mpls**: Sets MPLS specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets the QoS Group.
		• **traffic-class**: Sets the traffic class.

	Command or Action	Purpose

Step 7	**end**
	Example:
	`Device(config-pmap-c) #end`
	`Device(config-pmap-c) #`
		Saves the configuration changes.

What to do next

Configure the interface.

Creating a Traffic Queueing Policy

To create a traffic queueing policy for QoS, peform this procedure.

The following queueing policy map class-actions are supported:

- bandwidth: Bandwidth configuration options.
- exit: Exits from the QoS class action configuration mode.
- no: Negates or sets default values for the command.
- priority: Configures strict scheduling priority for the class.
- queue-limit: Queue threshold for tail drop configuration options.
- random-detect: Enables Random Early Detection as drop policy.
- service-policy: Configures QoS service policy.
- shape: Traffic shaping configuration options.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing test`
	`Device(config-pmap)#`
		Specifies the name of the queueing profile policy and enters policy map configuration mode.

Step 3	**class** *class name* **Example:**	Specifies the name of the class to be associated with the policy and enters policy class map

	Command or Action	Purpose
	`Device(config-pmap)# class traffic-class7` `Device(config-pmap-c)#`	configuration mode. Command options for policy class map configuration mode include the following: • *word*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth remaining ratio** *ratio*
	Example:
	`Device(config-pmap)# bandwidth remaining ratio 10`
	`Device(config-pmap-c)#`
		(Optional) Configures ratio for sharing excess bandwidth. The range is from 1 to 100.
		Note **bandwidth remaining ratio** and **priority level** cannot be configured simultaneously for the same class, even though they can be in the same policy.

Step 5	**priority level** *level*
	Example:
	`Device(config-pmap)# priority level 1`
	`Device(config-pmap-c)#`
		(Optional) Configures multi-level priority queue. The range is from 1 to 7.
		Note Priority level 1 can be configured only for traffic-class 7, priority level 2 can be configured only for traffic-class 6, and so on. Each traffic-class value can be mapped only to its respective priority level only.

Step 6	**queue-limit** *value* [**bytes**]
	Example:
	`Device(config-pmap-c)# queue-limit 100000 bytes`
	`Device(config-pmap-c)#`
		(Optional) Sets the queue limit threshold percentage value in bytes. The range is from 1000000 to 396000000.

Step 7	**random-detect** {**discard-class** | **discard-class-based** | **exponential-weighting-constant**}
	Example:
	`Device(config-pmap)# random-detect discard-class-based`
	`Device(config-pmap-c)#`
		(Optional) Enables Random Early Detection as the drop policy.
		• **discard-class**: Parameters for each discard-class value (0 or 1).
		Note **discard-class** and **priority** cannnot be configured simultaneously for the same class, even though they can be in the same policy.

	Command or Action	Purpose
		• **discard-class-based**: Enables discard-class-based WRED as the drop policy. • **exponential-weighting-constant**: Weight for mean queue depth calculation (0 to 15).

Step 8	**service-policy** *name*
	Example:
	`Device(config-pmap)# service-policy policy1`
	`Device(config-pmap-c)#`
		(Optional) Configures QoS Service Policy.
		Note policy1 used in this example should be a queueing policy. It can be configured only under class-default.

Step 9	**shape average** {*Kb/s* | **percent**}
	Example:
	`Device(config-pmap-c)# shape average 1000000000`
	`Device(config-pmap-c)#`
		Configures the traffic shaping average. The parameters include:
		• *Kb/s*: Use this command to configure a specific value. The range is 1.2 Mbps to 400 Gbps.
		• **percent**: Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is divided along all non priority queues based on bandwidth remaining ratio.

Step 10	**end** **Example:** `Device(config-cmap)# end`	Exits class map configuration mode and returns to privileged EXEC mode.

Configuring Class-Based Packet Marking for Ingress Policy-map

This procedure explains how to configure the following class-based packet marking features on your device:

- CoS value
- DSCP value
- IP value
- Precedence value
- QoS group value

- Traffic-class value
- Discard-class value
- MPLS experimental value

Before you begin

You should have created a class map and a policy map before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy1`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class-default`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change.
		Command options for policy class map configuration mode include the following:
		• **exit**: Exits from the QoS class action configuration mode.
		• **no**: Negates or sets default values for the command.
		• **police**: Policer configuration options.
		• **service-policy**: Configures the QoS service policy.
		• **set**: Sets QoS values using the following options:
		  • CoS values
		  • DSCP values
		  • Precedence values
		  • QoS group values

	Command or Action	Purpose
		Note This procedure describes the supported configurations using **set** command options. The other command options (**bandwidth**) are described in other sections of this guide. Only one **set** command for QoS tag (CoS/DSCP/PREC/EXP) is supported per class. The same class can also have set traffic-class and/or set discard-class if it is a ingress policy.

Step 4	**set cos** {*cos value* | **cos table** *table-map name* | **qos-group table** *table-map name*}
	Example:
	`Device(config-pmap)# set cos cos table cos-mapping`
	`Device(config-pmap)#`
		(Optional) Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.
		You can also set the CoS values using a table map. The following is a list of from types supported when using a table map. For example, **set cos qos-group table** *table-map name* means use the table-map to map from QoS-group to CoS.
		• **cos table**: Sets the CoS value based on a table map.
		• **qos-group table**: Sets the CoS value from QoS group based on a table map.
		Note **set cos qos-group table** is only supported in egress policy.
		Note Table-map is only supported in class-default.

Step 5	**set dscp** {*dscp value* | **default** | **dscp table** *table-map name* | **ef** | **precedence table** *table-map name* | **qos-group table** *table-map name* | **traffic-class table** *table-map name*}
	Example:
	`Device(config-pmap)# set dscp af11`
	`Device(config-pmap)#`
		(Optional) Sets the DSCP value.
		You can also set the DSCP values using a table map. The following is a list of from types supported when using a table map. For example, **set dscp qos-group table** *table-map name* means use the table-map to map from QoS-group to DSCP.
		• **dscp table**: Sets the DSCP value from DSCP based on a table map.
		• **precedence table**: Sets the DSCP value from precedence based on a table map.

	Command or Action	Purpose
		• **qos-group table**: Sets the DSCP value from a QoS group based upon a table map. **Note** Table-map is only supported in class-default.

Step 6	**set ip** {**dscp** | **precedence**}
	
	Example:
	
	`Device(config-pmap)# set ip dscp c3`
	`Device(config-pmap)#`
		(Optional) Sets IP specific values. These values are either IP DSCP or IP precedence values.
		
		You can also set the IP values using a table map. The following is a list of from types supported when using a table map. For example, **set ip dscp qos-group table** *table-map name* means use the table-map to map from QoS-group to IP DSCP.
		
		Note Table-map is only supported in class-default.
		
		You can set the following values using the **set ip dscp** command:
		
		• **dscp table**: Sets the DSCP value from DSCP based on a table map.
		
		• **precedence table**: Sets the DSCP value from precedence based on a table map.
		
		• **qos-group table**: Sets the DSCP value from a QoS group based upon a table map.
		
		The following is a list of from types supported when using a table map. For example, **set ip precedence qos-group table** *table-map name* means use the table-map to map from QoS-group to IP precedence.
		
		• *precedence value*: Sets the precedence value (from 0 to 7) .
		
		• **dscp table**: Sets the precedence value from DSCP value based on a table map.
		
		• **precedence table**: Sets the precedence value from precedence based on a table map
		
		• **qos-group table**: Sets the precedence value from a QoS group based upon a table map.

	Command or Action	Purpose
		Note **set ip precedence qos-group table** *table-map name* is only supported in egress.

Step 7	**set precedence** {*precedence value* | **dscp table** *table-map name* | **precedence table** *table-map name* | **qos-group table** *table-map name* | **traffic-class table** *table-map name*}
	Example:
	`Device(config-pmap)# set precedence 5`
	`Device(config-pmap)#`
		(Optional) Sets precedence values in IPv4 and IPv6 packets.
		You can also set the precedence values using a table map. The following is a list of from types supported when using a table map. For example, **set precedence qos-group table** *table-map name* means use the table-map to map from QoS-group to precedence.
		• *precedence value*: Sets the precedence value (from 0 to 7) .
		• **dscp table**: Sets the precedence value from DSCP value based on a table map.
		• **precedence table**: Sets the precedence value from precedence based on a table map.
		• **qos-group table**: Sets the precedence value from a QoS group based upon a table map.
		Note Table-map is only supported in class-default.

Step 8	**set discard-class** {*discard-class value* | **cos table** *table-map name* | **dscp table** *table-map name* | **mpls experimental table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set discard-class 0`
	`Device(config-pmap)#`
		(Optional) Sets discard-class values. You can also set the discard-class values using a table map.
		Note Table-map is only supported in class-default.
		• **cos table**: Sets the discard-class value from CoS value based on a table map.
		• **dscp table**: Sets the discard-class value from DSCP value based on a table map.
		• **mpls experimental table**: Sets the discard-class value from MPLS experimental based on a table map.
		• **precedence table**: Sets the discard-class value from precedence value based on a table map.

	Command or Action	Purpose

Step 9	**set qos-group** {*qos-group value* | **dscp table** *table-map name* | **mpls experimental table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set qos-group 10`
	`Device(config-pmap)#`
		(Optional) Sets QoS group values. You can also set the QoS group using a table map.
		• **cos table**: Sets QoS group from CoS value based on a table map.
		• **dscp table**: Sets QoS group from DSCP value based on a table map.
		• **mpls experimental table**: Sets QoS group from MPLS experimental value based on a table map.
		• **precedence table**: Sets QoS group from precedence based on a table map.

Step 10	**set traffic-class** {*traffic-class value* | **cos table** *table-map name* | **dscp table** *table-map name* | **mpls experimental table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set traffic-class 3`
	`Device(config-pmap)#`
		(Optional) Sets traffic-class values from 0 to 7, or from the following table-maps:
		• **cos table**: Sets the traffic-class value from COS value based on a table map.
		• **dscp table**: Sets the traffic-class value from DSCP value based on a table map.
		• **mpls experimental table**: Sets the traffic-class value from MPLS experimental value based on a table map.
		• **precedence table**: Sets the traffic-class value from precedence value based on a table map.

Step 11	**end**
	Example:
	`Device(config-pmap)# end`
	`Device#`
		Exits policy map configuration mode and returns to privileged EXEC mode.

Step 12	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Attach the traffic policy to an interface using the **service-policy** command.

Attaching a Traffic Policy to an Interface

After the traffic class and traffic policy are created, you must use the **service-policy** interface configuration command to attach a traffic policy to an interface, and to specify the direction in which the policy should be applied (either on packets coming into the interface or packets leaving the interface).

Before you begin

A traffic class and traffic policy must be created before attaching a traffic policy to an interface.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface** *type*
	Example:
	`Device(config)# interface fortygigabitEthernet1/0/1`
	`Device(config-if)#`
		Enters interface configuration mode and configures an interface.
		Command parameters for the interface configuration include:
		• **TenGigabitEthernet**: 10-Gigabit Ethernet
		• **TwentyFiveGigabitEthernet**: 25-Gigabit Ethernet
		• **FortyGigabitEthernet**: 40-Gigabit Ethernet
		• **HundredGigabitEthernet**: 100-Gigabit Ethernet
		Note • Tunnel and Vlan interface are not supported.
		• Policing, queuing, and marking are not supported on the management interface.

Step 3	**service-policy** [**type queueing**] {**input** *policy-map* | **output** *policy-map*} **Example:** `Device(config-if)# service-policy output policy_map_01`	Attaches a policy map to an input or output interface. This policy map is then used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface (egress).

	Command or Action	Purpose
	`Device(config-if)#`	**Note** • Non-queuing policy with policer is supported only in ingress. • On egress, only non-queueing policy with marking is supported. • Queuing policy (**type queueing**) is supported only in egress.

Step 4	**end**
	Example:
	`Device(config-if)# end`
	`Device#`
		Exits interface configuration mode and returns to privileged EXEC mode.

Step 5	**show policy map** **Example:** `Device# show policy map`	(Optional) Displays statistics for the policy on the specified interface.

What to do next

Proceed to attach any other traffic policy to an interface, and to specify the direction in which the policy should be applied.

Classifying, Policing, and Marking Traffic on Physical Ports by Using Policy Maps

You can configure a nonhierarchical policy map on a physical port that specifies which traffic class to act on. Actions supported are remarking and policing.

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose

Step 2	**class-map** {*class-map name* | **match-any** | **match-all**}
	Example:
	`Device(config)# class-map ipclass1`
	`Device(config-cmap)# exit`
	`Device(config)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		• If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** { *access list index* | *access list name* }
	Example:
	`Device(config-cmap)# match access-group 1000`
	`Device(config-cmap)# exit`
	`Device(config)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• ip
		• precedence
		• qos-group
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map ipclass1`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

Step 5	**class** {*class-map-name* | **class-default**} **Example:**	Defines a traffic classification, and enter policy-map class configuration mode.

	Command or Action	Purpose
	`Device(config-pmap)# class ipclass1` `Device(config-pmap-c)#`	By default, no policy map class-maps are defined. If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command. A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 6	**set** {**cos** | **discard-class** | **dscp** | **ip** | **mpls** | **precedence** | **qos-group** | **traffic-class**}
	Example:
	`Device(config-pmap-c)# set dscp 45`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **discard-class**: Sets discard behavior identifier.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **mpls**: Sets MPLS specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets the QoS Group.
		• **traffic-class**: Sets the traffic class.
		In this example, the **set dscp** command classifies the IP traffic by setting a new DSCP value in the packet.

Step 7	**police** {*target_bit_rate* | **cir** | **rate** }
	Example:
	`Device(config-pmap-c)# police 100000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*: Specifies the bit rate per second, enter a value between 8000 and 400000000000.
		• **cir**: Committed Information Rate.
		• **rate**: Specifies the police rate PCR for hierarchical policies.

	Command or Action	Purpose
		In this example, the **police** command adds a policer to the class where any traffic beyond the 100000 set target bit rate is dropped.
Step 8	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 9	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.
Step 10	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigabitEthernet 1/0/2`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 11	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input ipclass1`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map ipclass1`	(Optional) Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

If applicable to your QoS configuration, configure classification, policing, and marking of traffic on SVIs by using policy maps.

Classifying and Marking Traffic by Using Policy Maps

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by using policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** {*class-map name* | **match-any** | **match-all**}
	Example:
	`Device(config)# class-map class_cos2`
	`Device(config-cmap)# match cos 2`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		• If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**policy-map** *policy-map-name*
	Example:
	`Device(config-cmap)# policy-map policy_cos2`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

	Command or Action	Purpose
Step 4	**description** *description* **Example:** `Device(config-pmap)# description cos 2`	(Optional) Enters a description of the policy map.

Step 5	**class** {*class-map-name* | **class-default**}
	Example:
	`Device(config-pmap)# class class_cos2`
	`Device(config-pmap-c)#`
		Defines a traffic classification, and enters the policy-map class configuration mode.
		By default, no policy map class-maps are defined.
		If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command.
		A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 6	**set** {**cos** | **qos-group** | **traffic-class** | **discard-class**}
	Example:
	`Device(config-pmap-c)# set cos 7`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **qos-group**: Sets QoS group.
		• **traffic-class**: Sets traffic class.
		• **discard-class**: Sets discard class.
		In this example, the **set cos** command classifies the traffic by matching the packets with CoS values of 2, and sets the CoS value to a new value of 7.

Step 7	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 8	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 9	**interface** *interface-id* **Example:** `Device(config)# interface hundredgigabitethernet 1/0/3`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 10	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input policy_cos2`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 11	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 12	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map`	(Optional) Verifies your entries.
Step 13	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Table Maps

Table maps are a form of marking, and also enable the mapping and conversion of one field to another using a table. For example, a table map can be used to map and convert a Layer 2 CoS setting to a new or same CoS value, and also convert a Layer 2 CoS setting to a traffic class or a QoS group. Similar settings apply to Layer 3 DSCP and Precedence.

When setting a table-map in a policy, the **from** and **to** values should be of the same type, example DCSP to DSCP, COS to COS, and so on. This exception is not applicable for QoS-group and traffic-class where **from** and **to** values are of different types, for example, DSCP to traffic-class, COS to traffic-class, and so on.

Note

- A table map can be referenced in multiple policies or multiple times in the same policy.
- The examples in this procedure uses table maps to map DSCP value to traffic-class.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**table-map** *name* {**default** {*default value* | **copy** | **ignore**} | **exit** | **map** {**from** *from value* **to** *to value* } | **no**}
	Example:
	`Device(config)# table-map table01`
	`Device(config-tablemap)#`
		Creates a table map and enters the table map configuration mode. In table map configuration mode, you can perform the following tasks:
		• **default**: Configures the table-map. The default behaviour is 'default copy', which can be set or modified to default ignore or default x, where x can be CoS or DSCP or traffic-class or QoS-group or EXP values.
		• **exit**: Exits from the table map configuration mode.
		• **map**: Maps a *from* to a *to* value in the table map.
		• **no**: Negates or sets the default values of the command.

Step 3	**map from** *value* **to** *value*
	Example:
	`Device(config-tablemap)# map from 0 to 2`
	`Device(config-tablemap)# map from 1 to 4`
	`Device(config-tablemap)# map from 24 to 3`
	`Device(config-tablemap)# map from 40 to 6`
	`Device(config-tablemap)# default 0`
	`Device(config-tablemap)#`
		In this step, packets with DSCP value 0 are marked to the traffic-class 2, DSCP value 1 to the traffic-class 4, DSCP value 24 to the traffic-class 3, DSCP value 40 to the traffic-class 6 and all others to the traffic-class 0.
		Note The mapping from DSCP to traffic-class values in this example is configured by using the **set** policy map class configuration command as described in a later step in this procedure.

Step 4	**exit**
	Example:
	`Device(config-tablemap)# exit`
	`Device(config)#`
		Returns to global configuration mode.

Step 5	**exit**	Returns to privileged EXEC mode.

	Command or Action	Purpose

	Example:
	`Device(config) exit`
	`Device#`	

Step 6	**show table-map**
	Example:
	`Device# show table-map`
	`Table Map table01`
	`from 0 to 2`
	`from 1 to 4`
	`from 24 to 3`
	`from 40 to 6`
	`default 0`
		Displays the table map configuration.

Step 7	**configure terminal**
	Example:
	`Device# configure terminal`
	`Device(config)#`
		Enters global configuration mode.

Step 8	**policy-map**
	Example:
	`Device(config)# policy-map table-policy`
	`Device(config-pmap)#`
		Configures the policy map for the table map.

Step 9	**class class-default**
	Example:
	`Device(config-pmap)# class class-default`
	`Device(config-pmap-c)#`
		Matches the class to the system default.

Step 10	**set traffic-class dscp table** *table map name*
	Example:
	`Device(config-pmap-c)# set traffic-class dscp table table01`
	`Device(config-pmap-c)#`
		Incoming packets with DSCP value are mapped to hit a particular VOQ which is defined by the traffic-class values in a table-map.

Step 11	**end** **Example:** `Device(config-pmap-c)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device#`	

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

How to Configure QoS Features and Functionality

The following sections provide configurational information about QoS features and functionality.

Configuring Bandwidth

This procedure explains how to configure bandwidth on your device.

Before you begin

You should have created a created a class-map for bandwidth based on traffic-class classification before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing policy_bandwidth01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class traffic-class7`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

	Command or Action	Purpose

Step 4	**bandwidth remaining ratio** *ratio*
	Example:
	`Device(config-pmap-c)# bandwidth remaining ratio 10`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map.
		• *ratio*: Ratios can range from 1 to 63.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating the policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Police

This procedure explains how to configure policing on your device.

Before you begin

You should have created a class map for policing before beginning this procedure.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`Device# configure terminal`
	`Device(config)#`
		Enters global configuration mode.

Step 2	**policy-map** *policy name* **Example:** `Device(config)# policy-map policy_police01`	Enters policy map configuration mode. Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose
	`Device(config-pmap)#`	

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class_police01`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**police** {*target_bit_rate* [**bc** | **conform-action** | **pir**] | **cir** {*target_bit_rate* | **percent** *percentage*} | **rate** {*target_bit_rate* | **percent** *percentage*} **conform-action transmit exceed-action** {**drop** [**violate action**] | **set-cos-transmit** | **set-dscp-transmit** | **set-prec-transmit** | **transmit** [**violate action**] }}
	Example:
	`Device(config-pmap-c)# police 1200000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		The following **police** subcommand options are available:
		• *target_bit_rate*: Bits per second (from 1200000 to 400000000000).
		• **bc**: Conform burst.
		• **conform-action**: Action taken when rate is less than conform burst.
		• **pir**: Peak Information Rate.
		• **cir**: Committed Information Rate.
		• *target_bit_rate*: Target bit rate (from 1200000 to 400000000000).
		• **percent**: Percentage of interface bandwidth for CIR.
		• **rate**: Specifies the police rate, PCR for hierarchical policies, or SCR for single-level ATM 4.0 policer policies.
		• *target_bit_rate*: Target Bit Rate (from from 1200000 to 400000000000).
		• **percent**: Percentage of interface bandwidth for rate.
		The following **police conform-action transmit exceed-action** subcommand options are available:
		• **drop**: Drops the packet.
		• **set-cos-transmit**: Sets the CoS value and sends it.

	Command or Action	Purpose
		• **set-discard-class-transmit**: Sets the discard-class value and sends it. • **set-dscp-transmit**: Sets the DSCP value and sends it. • **set-prec-transmit**: Rewrites the packet precedence and sends it. • **transmit**: Transmits the packet. **Note** Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies. **Note** The **show policy-map** command output does not display counters for conformed bytes and exceeded bytes

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Priority

This procedure explains how to configure priority on your device.

Note The device supports giving priority to specified queues. There are seven priority levels available (1 to 7).

Before you begin

You should have created a class map for priority before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing policy_priority01`
	`Device(config-pmap)#`
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy, and enters policy map configuration mode.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class traffic-class7`
	`Device(config-pmap-c)#`
		Specifies the name of the class whose policy you want to create or change, and enters policy class map configuration mode. Command options for policy class map configuration mode include the following:
		• *class name*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

Step 4	**priority level** *level_value*
	Example:
	`Device(config-pmap-c)# priority level 1`
	`Device(config-pmap-c)#`
		(Optional) This command assigns a strict scheduling priority for the class.
		• *level_value*: Specifies the multilevel (1-7) priority queue.
		Note Priority level 1 is the highest priority level, followed by priority levels 2, 3, 4, 5, 6, and 7 respectively. Queues mapped to high priority classes are serviced before queues mapped to lower priority classes.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

	Command or Action	Purpose
Step 6	**show policy-map type queueing** **Example:** `Device# show policy-map type queueing`	(Optional) Displays the runtime representation and statistics of all the queueing policies configured on the device.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Queues and Shaping

The following sections provide configurational information about queueing and shaping.

Configuring Egress Queue Characteristics

Depending on the complexity of your network and your QoS solution, you may need to perform all of the procedures in this section. You need to make decisions about these characteristics:

- Which packets are mapped by DSCP, CoS, or QoS group value to each queue and threshold ID?
- Which traffic class based classification apply to the queues, and if incoming packets are mapped to any of the traffic classes from 0 to 7, so that traffic gets queued in specific VoQ queues?
- How much of the fixed buffer space is allocated to the queues?
- Does the bandwidth of the port need to be rate limited?
- How often should the egress queues be serviced and which technique (shaped, shared, or both) should be used?

Note You can only configure the egress queues on the device, and only traffic-class based classification is supported.

Configuring Queue Limits

Queue-limit can be only be configured in unit of bytes, and total number of thresholds are limited.

Before you begin

The following are prerequisites for this procedure:

- You should have created a class map for the queue limits before beginning this procedure.
- You must have configured either bandwidth, shape, or priority on the policy map prior to configuring the queue limits.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queuing** *policy name*
	Example:
	`Device(config)# policy-map type queueing test`
	`Device(config-pmap)#`
		Specifies the name of the queueing profile policy and enters policy map configuration mode.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class traffic-class7`
	`Device(config-pmap-c)#`
		Specifies the name of the class to be associated with the policy and enters policy class map configuration mode. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

Step 4	**shape average** {*bits/s* | **percent**}
	Example:
	`Device(config-pmap-c)# shape average 1000000000`
	`Device(config-pmap-c)#`
		Configures the traffic shaping average. The parameters include:
		• *bits/s*: Use this command to configure a specific value. The range is 1200000 to 400000000000.
		• **percent**: Allocates a maximum bandwidth to a particular class. The value can be from 1 to 100. Shaping is done on the percentage value, and traffic will not exceed the shape percentage value.

Step 5	**queue-limit** *value* [**bytes**] **Example:** `Device(config-pmap-c)# queue-limit 100000 bytes`	Sets the queue limit threshold percentage value in bytes. The range is from 1000000 to 396000000.
Step 6	**end** **Example:**	Exits policy class map configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-pmap-c)# end` `Device#`	

Configuring Shaping

You use the **shape** command to configure shaping (maximum bandwidth) for a particular class. The queue's bandwidth is restricted to this value even though the port has additional bandwidth left. You can configure shaping as an average percent, as well as a shape average value in bits per second.

Before you begin

You should have created a class map for shaping before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queuing** *policy name*
	Example:
	`Device(config)# policy-map type queuing policy_shaping01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class traffic-class7`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *class name*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

Step 4	**shape average** {*target bit rate* | **percent** *percentage*} **Example:** `Device(config-pmap-c)# shape average percent 50`	Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

	Command or Action	Purpose
	`Device(config-pmap-c)#`	

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Sharped Profile Queuing

This procedure explains how to configure sharped profile queuing on your switch:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing policy_shaping01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
		policy name is the name of the child policy map. The name can be a maximum of 40 alphanumeric characters.

Step 3	**class** class name
	Example:
	`Device(config-pmap)# class traffic-class1`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *class name*: Class map name.

	Command or Action	Purpose
		• **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth remaining ratio** *ratio*
	Example:
	`Device(config-pmap)# bandwidth remaining ratio 10`
	`Device(config-pmap-c)#`
		(Optional) Configures ratio for sharing excess bandwidth. The range is from 1 to 63.

Step 5	**shape average** {*target bit rate* | **percent** *percentage*}
	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Exits policy class map configuration mode and returns to privileged EXEC mode.

Sharped Profile Queuing Configuration

The following is a sample output for sharped queuing:

```
Device# show policy-map type queueing parent class class-default

   Class class-default
      Average Rate Traffic Shaping
      cir 5%
  Policy Map type queueing child
    Class tc7
      priority level 1
      Average Rate Traffic Shaping
      cir 1500000000 (bps)
    Class class-default
```

Monitoring QoS

The following commands can be used to monitor QoS on the device:

Table 24: Monitoring QoS

Command	Description
show class-map [*class_map_name*]	Displays a list of all class maps configured.
show policy-map [**type queueing**] [*policy_map_name*]	Displays a list of all policy maps configured. Command parameters include: • policy map name • type queueing
show policy-map interface {**TenGigabitEthernet** | **TwentyfiveGigabitEthernet** | **FortyGigabitEthernet** | **HundredGigabitEthernet**}	Displays the runtime representation and statistics of all the policies configured on the device. Command parameters include: • TenGigabitEthernet: 10-Gigabit Ethernet • TwentyFiveGigabitEthernet: 25-Gigabit Ethernet • FortyGigabitEthernet: 40-Gigabit Ethernet • HundredGigabitEthernet: 100-Gigabit Ethernet **Note** Though **wireless** option is visible on the CLI, it is not supported.

Command	Description
show policy-map type queueing interface {**TenGigabitEthernet** | **TwentyfiveGigabitEthernet** | **FortyGigabitEthernet** | **HundredGigabitEthernet**}	Displays the runtime representation and statistics of all the queueing policies configured on the device. Command parameters include: • TenGigabitEthernet: 10-Gigabit Ethernet • TwentyFiveGigabitEthernet: 25-Gigabit Ethernet • FortyGigabitEthernet: 40-Gigabit Ethernet • HundredGigabitEthernet: 100-Gigabit Ethernet **Note** Though **wireless** option is visible on the CLI, it is not supported.
show table-map	Displays all the table maps and their configurations.

Configuration Examples for QoS

The following sections provide configuration examples for QoS.

Examples: TCP Protocol Classification

TCP packets can be classified based on port numbers. The configuration for TCP protocol is as follows:

```
Device# show ip acce tcp

Extended IP access list tcp
    10 permit tcp any any eq 80
Device #
Device #show run class-map tcp

Current configuration : 63 bytes
!
class-map match-all tcp
 match access-group name tcp
!
end

Device# show run policy-map tcp

Current configuration : 56 bytes
!
policy-map tcp
```

```
 class tcp
  police 1000000000
!
end

Device# show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy input tcp
end

Device #
```

Examples: UDP Protocol Classification

UDP packets can be classified based on port numbers. The configuration example for UDP protocol is as follows:

```
Device# show ip acce udp

Extended IP access list udp
    10 permit udp any any eq ntp
Device #

Device# show run class-map udp
Building configuration...

Current configuration : 63 bytes
!
class-map match-all udp
 match access-group name udp
!
end

Device# show run policy-map udp
Building configuration...

Current configuration : 56 bytes
!
policy-map udp
 class udp
  police 1000000000
!
end

Device# show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy input udp
end

Device #
```

Examples: RTP Protocol Classification

RTP packets can be classified based on port numbers. The configuration example for RTP protocol is as follows:

```
Device# show ip access-list rtp

Extended IP access list rtp
    10 permit udp any any eq 554
    11 permit tcp any any eq 554
Device #

Device# show run class-map rtp

Current configuration : 63 bytes
!
class-map match-all rtp
 match access-group name rtp
!
end

Device# show run policy-map rtp

Current configuration : 56 bytes
!
policy-map rtp
 class rtp
  police 1000000000
!
end

Device# show run int tw 1/0/1

Current configuration : 93 bytes
!
interface TwentyFiveGigE1/0/1
 no ip address
 no keepalive
 service-policy input rtp
end

Device #
```

Examples: Classification by Access Control Lists

This example shows how to classify packets for QoS by using access control lists (ACLs):

```
Device# configure terminal
Device(config)# access-list 101 permit ip host 12.4.1.1 host 15.2.1.1
Device(config)# class-map acl-101
Device(config-cmap)# description match on access-list 101
Device(config-cmap)# match access-group 101
Device(config-cmap)#
```

After creating a class map by using an ACL, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service Layer 2 Classification

This example shows how to classify packets for QoS using a class of service Layer 2 classification:

```
Device# configure terminal
Device(config)# class-map match-any cos
Device(config-cmap)# match cos ?
  <0-7>  Enter up to 4 class-of-service values separated by white-spaces
Device(config-cmap)# match cos 3 4 5
Device(config-cmap)#
```

After creating a class map by using a CoS Layer 2 classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service DSCP Classification

This example shows how to classify packets for QoS using a class of service DSCP classification:

```
Device# configure terminal
Device(config)# class-map match-any dscp
Device(config-cmap)# match dscp af21 af22 af23
Device(config-cmap)#
```

After creating a class map by using a DSCP classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Classification by DSCP or Precedence Values

This example shows how to classify packets by using DSCP or precedence values:

```
Device# configure terminal
Device(config)# class-map prec2
Device(config-cmap)# description matching precedence 2 packets
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit
Device(config)# class-map ef
Device(config-cmap)# description EF traffic
Device(config-cmap)# match ip dscp ef
Device(config-cmap)# exit

Device(config)# class-map prec5
Device(config-cmap)# match precedence 5
Device(config-cmap)# exit

Device(config)# class-map cs1
Device(config-cmap)# match dscp cs1
Device(config-cmap)#
```

After creating a class map by using a DSCP or precedence values, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Hierarchical Policy Configuration

The following is an example of non-queueing hierarchical policy configuration:

Note For hierarchical policy, overlapping actions are not supported. For example, a policer in both parent and child are not supported, and marking in both parent and child are not supported.

```
Device(config)# class-map dscp8
Device(config-cmap)# match dscp 8
Device(config-cmap)# exit

Device(config)# class-map pre2
Device(config-cmap)# match pre 2
Device(config-cmap)# end

Device(config)# policy-map
Device(config)# policy-map child
Device(config-pmap)# class dscp8
Device(config-pmap-c)# set dscp af11
Device(config-pmap-c)# class pre2
Device(config-pmap-c)# set precedence 5
Device(config-pmap-c)# !
Device(config-pmap-c)# end

Device(config)# policy-map parent
Device(config-pmap)# class class-default
Device(config-pmap-c)# police rate percent 20
Device(config-pmap-c-police)# service-policy child
Device(config-pmap-c)# !
Device(config-pmap-c)# end
```

The following is an example of queueing hierarchical policy configuration:

```
Device(config)# policy-map type queueing child_queue
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average percent 10
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# class traffic-class6
Device(config-pmap-c)# shape average percent 20
Device(config-pmap-c)# bandwidth remaining ratio 10
Device(config-pmap-c)# !
Device(config-pmap-c)# end

Device(config)# policy-map type queueing parent_queue
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average percent 80
Device(config-pmap-c)# service-policy child_queue
Device(config-pmap-c)# !
Device(config-pmap-c)# end
```

The following is an example of a non-queueing hierarchical policy configuration using table maps:

```
Device(config)# table-map dscp2dscp
Device(config-tablemap)# default copy
Device(config)# exit

Device(config)# policy-map child_policy
Device(config-pmap)# class dscp8
Device(config-pmap-c)# police rate 1000000000
Device(config-pmap-c-police)# class pre2
```

```
Device(config-pmap-c)# police rate 2000000000
Device(config-pmap-c-police)# !
Device(config-pmap-c-police)# end

Device(config)# policy-map parent_policy
Device(config-pmap)# class class-default
Device(config-pmap-c)# set dscp dscp table dscp2dscp
Device(config-pmap-c)# service-policy child_policy
Device(config-pmap-c)# !
Device(config-pmap-c)# end
```

Examples: Classification for Voice and Video

This example describes how to classify packet streams for voice and video using device specific information.

In this example, voice and video are coming in from end-point A into HundredGigabitEthernet1/0/1on the device and have precedence values of 5 and 6, respectively. Additionally, voice and video are also coming from end-point B into FortyGigabitEthernet1/0/2 on the device with DSCP values of EF and AF11, respectively.

Assume that all the packets from the both the interfaces are sent on the uplink interface, and there is a requirement to police voice to 100 Mbps and video to 150 Mbps.

To classify per the above requirements, a class to match voice packets coming in on HundredGigabitEthernet1/0/1 is created, named voice-interface-1, which matches precedence 5. Similarly another class for voice is created, named voice-interface-2, which will match voice packets in HundredGigabitEthernet1/0/3. These classes are associated to two separate policies named input-interface-1, which is attached to HundredGigabitEthernet1/0/1, and input-interface-2, which is attached to HundredGigabitEthernet1/0/3. The action for this class is to mark the qos-group to 10. To match packets with QoS-group 10 on the output interface, a class named voice is created which matches on QoS-group 10. This is then associated to another policy named output-interface, which is associated to the uplink interface. Video is handled in the same way, but matches on QoS-group 20.

The following example shows how classify using the above device specific information:

```
Device(config-cmap)# class-map voice-interface-1
Device(config-cmap)# match ip precedence 5
Device(config-cmap)# exit

Device(config)# class-map video-interface-1
Device(config-cmap)# match ip precedence 6
Device(config-cmap)# exit

Device(config)# class-map voice-interface-2
Device(config-cmap)# match ip dscp ef
Device(config-cmap)# exit

Device(config)# class-map video-interface-2
Device(config-cmap)# match ip dscp af11
Device(config-cmap)# exit

Device(config)# policy-map input-interface-1
Device(config-pmap)# class voice-interface-1
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# set traffic-class 7

Device(config-pmap-c)# class video-interface-1
Device(config-pmap-c)# set qos-group 20
Device(config-pmap-c)# set traffic-class 6
```

```
Device(config-pmap-c)# policy-map input-interface-2
Device(config-pmap)# class voice-interface-2
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# set traffic-class 7
Device(config-pmap-c)# class video-interface-2
Device(config-pmap-c)# set qos-group 20
Device(config-pmap-c)# set traffic-class 6

Device(config)# class-map match-all traffic-class7
Device(config-cmap)# match traffic-class 7
Device(config-cmap)# exit

Device(config)# class-map match-all traffic-class6
Device(config-cmap)# match traffic-class 6

Device(config)# policy-map type queueing output-interface
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average 2g
Device(config-pmap-c)# class traffic-class6
Device(config-pmap-c)# shape average 1g
Device(config-pmap-c)# end
```

Examples: Queue-limit Configuration

The following example shows how to configure a queue-limit policy based on per traffic-class:

```
Device# configure terminal
Device#(config)# policy-map type queueing test
Device#(config-pmap)# class tc7
Device#(config-pmap-c)# shape average 1000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc6
Device#(config-pmap-c)# shape average 2000000000
Device#(config-pmap-c)# queue-limit 2000000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc5
Device#(config-pmap-c)# shape average 3000000000
Device#(config-pmap-c)# queue-limit 3000000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc4
Device#(config-pmap-c)# shape average 4000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc3
Device#(config-pmap-c)# shape average 5000000000
Device#(config-pmap-c)# queue-limit 200000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc2
Device#(config-pmap-c)# shape average 4000000000
Device#(config-pmap-c)# queue-limit 300000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc1
Device#(config-pmap-c)# shape average 3000000000
```

```
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class class-default
Device#(config-pmap-c)# shape average 2000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# end
Device#
```

Examples: Policing Action Configuration

The following example displays the various policing actions that can be associated to the policer. These actions are accomplished using the conforming, exceeding, or violating packet configurations. You have the flexibility to drop, mark and transmit, or transmit packets that have exceeded or violated a traffic profile.

For example, a common deployment scenario is one where the enterprise customer polices traffic exiting the network towards the service provider and marks the conforming, exceeding and violating packets with different DSCP values. The service provider could then choose to drop the packets marked with the exceeded and violated DSCP values under cases of congestion, but may choose to transmit them when bandwidth is available.

Note The Layer 2 fields can be marked to include the CoS fields, and the Layer 3 fields can be marked to include the precedence and the DSCP fields.

One useful feature is the ability to associate multiple actions with an event. For example, you could set the precedence bit and the CoS for all conforming packets. A submode for an action configuration could then be provided by the policing feature.

This is an example of a policing action configuration:

```
Device# configure terminal
Device(config)# policy-map police
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 1000000 pir 2000000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# end
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Examples: Policing Units

The policing unit is the basis on which the token bucket works. CIR and PIR are specified in bits per second. The burst parameters are specified in bytes. This is the default mode; it is the unit that is assumed when no

units are specified. The CIR and PIR can also be configured in percent, in which case the burst parameters have to be configured in milliseconds.

The following is an example of a policer configuration in bits per second. In this configuration, a dual-rate three-color policer is configured where the units of measurement is bits. The burst and peak burst are all specified in bits.

```
Device(config)# policy-map bps-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police rate 1200000 peak-rate 12000000
conform-action transmit exceed-action set-dscp-transmit dscp table
DSCP_EXCE violate-action drop
```

Examples: Single-Rate Two-Color Policing Configuration

The following example shows how to configure a single-rate two-color policer:

```
Device(config)# class-map match-any prec1
Device(config-cmap)# match ip precedence 1
Device(config-cmap)# exit
Device(config)# policy-map policer
Device(config-pmap)# class prec1
Device(config-pmap-c)# police cir 256000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

Examples: Dual-Rate Three-Color Policing Configuration

The following example shows how to configure a dual-rate three-color policer:

```
Device# configure terminal
Device(config)# policy-Map dual-rate-3color-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 64000 bc 2000 pir 128000 be 2000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer based markdown actions are only supported using table maps.

Examples: Table Map Marking Configuration

The following steps and examples show how to use table map marking for your QoS configuration:

1. Define the table-map using the **table-map** command and indicate the mapping of the values. This table does not know of the policies or classes within which it will be used. The default command in the table map indicates the value to be copied into the 'to' field when there is no matching 'from' field. In the example, a table map named table-map1 is created. The mapping defined is to convert the value from 0 to 1 and from 2 to 3, while setting the default value to 4.

Note Only table-map values of same QoS tags are supported. For example, a table-map of type CoS to CoS or DSCP to DSCP or Prec to Prec are supported, and also a table-map of type DSCP/CoS/PREC to a QoS group and a table-map of type DSCP/CoS/PREC to a traffic-class.

```
Device(config)# table-map table-map1
Device(config-tablemap)# map from 0 to 1
Device(config-tablemap)# map from 2 to 3
Device(config-tablemap)# default 4
Device(config-tablemap)# exit
```

2. Define the policy map where the table map will be used.

   In the example, the incoming CoS is mapped to the CoS based on the mapping specified in the table table-map1. For this example, if the incoming packet has a CoS of 0, the CoS in the packet is set 1. If no table map name is specified the command assumes a default behavior where the value is copied as is from the 'from' field to the 'to' field.

```
Device(config)# policy-map policy1
Device(config-pmap)# class class-default
Device(config-pmap-c)# set cos cos table table-map1
Device(config-pmap-c)# exit
```

3. Associate the policy to an interface.

```
Device(config)# interface HundredGigabitE1/0/2
Device(config-if)# service-policy output policy1
Device(config-if)# exit
```

Displaying QoS Configuration

The following is a sample output of the **show policy-map** command:

```
Device# show policy-map mul4

  Policy Map mul4
    Class cs2
     police cir 1000000000 bc 1024000
       conform-action transmit
       exceed-action drop
      set dscp cs7
    Class cs3
      set traffic-class 7
     police cir 40000000000 bc 1024000
       conform-action transmit
```

```
    exceed-action drop
  Class cs1
   police cir 1000000000 bc 1024000
     conform-action transmit
     exceed-action drop
    set traffic-class 5
  Class cs5
   police cir 20000000000 bc 1024000
     conform-action transmit
     exceed-action drop
    set traffic-class 5
  Class cs6
   police cir 50000000 bc 10240
     conform-action transmit
     exceed-action drop
  Class dscp1
   police cir 5500000000 bc 1024000
     conform-action transmit
     exceed-action drop
    set traffic-class 2
    set dscp 45
  Class ttl
   police cir 1000000000 bc 1024000
     conform-action transmit
     exceed-action drop
  Class cs4
   police cir 10000000000 bc 1024000
     conform-action transmit
     exceed-action drop
    set traffic-class 5
  Class class-default
   police cir 1000000000 bc 1024000
     conform-action transmit
     exceed-action drop
```

The following is a sample output from the **show policy-map interface** command:

```
Device# show policy-map interface HundredGigE1/0/14

 HundredGigE1/0/14

  Service-policy input: mul4

    Class-map: cs2 (match-all)
      0 packets
      Match:  dscp cs2 (16)
      police:
          cir 1000000000 bps, bc 1024000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop
        conformed 0000 bps, exceeded 0000 bps
      QoS Set
        dscp cs7

    Class-map: cs3 (match-all)
      224757946122 packets
      Match:  dscp cs3 (24)
      QoS Set
        traffic-class 7
      police:
          cir 40000000000 bps, bc 1024000 bytes
        conformed 335092644777000 bytes; actions:
          transmit
```

```
    exceeded 2044274406000 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps

Class-map: cs1 (match-all)
  0 packets
  Match:  dscp cs1 (8)
  police:
      cir 1000000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps
  QoS Set
    traffic-class 5

Class-map: cs5 (match-all)
  0 packets
  Match:  dscp cs5 (40)
  police:
      cir 20000000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps
  QoS Set
    traffic-class 5

Class-map: cs6 (match-all)
  0 packets
  Match:  dscp cs6 (48)
  police:
      cir 50000000 bps, bc 10240 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps

Class-map: dscp1 (match-all)
  0 packets
  Match:  dscp 5
  police:
      cir 5500000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps
  QoS Set
    traffic-class 2
    dscp 45

Class-map: ttl (match-all)
  0 packets
  Match: access-group name ttl
  police:
      cir 1000000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
```

```
      conformed 0000 bps, exceeded 0000 bps

  Class-map: cs4 (match-all)
    0 packets
    Match:  dscp cs4 (32)
    police:
        cir 10000000000 bps, bc 1024000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps
    QoS Set
      traffic-class 5

  Class-map: class-default (match-any)
    5073 packets
    Match: any
    police:
        cir 1000000000 bps, bc 1024000 bytes
      conformed 1215000 bytes; actions:
        transmit
      exceeded 6394500 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps
```

The following is a sample output from the **show policy-map type queue interface** command. This show command displays information about which ingress packets are hitting which VoQs.

```
Device# show policy-map type queue interface HundredGigE1/0/14

 HundredGigE1/0/14

  Service-policy queueing output: llq

    queue stats for all priority classes:
      Queueing
      priority level 1
      queue limit 96000 bytes
      (total drops) 0
      (bytes output) 59924103953524

    queue stats for all priority classes:
      Queueing
      priority level 2
      queue limit 96000 bytes
      (total drops) 0
      (bytes output) 0

    queue stats for all priority classes:
      Queueing
      priority level 3
      queue limit 96000 bytes
      (total drops) 0
      (bytes output) 0

    queue stats for all priority classes:
      Queueing
      priority level 4
      queue limit 96000 bytes
      (total drops) 0
      (bytes output) 0

    queue stats for all priority classes:
```

```
  Queueing
  priority level 5
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

queue stats for all priority classes:
  Queueing
  priority level 6
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

queue stats for all priority classes:
  Queueing
  priority level 7
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

Class-map: tc7 (match-all)
  67482284685 packets
  Match: traffic-class 7
  shape (average) cir 5000000000, bc 20000000, be 20000000
  target shape rate 5000000000
  Priority: Strict,

  Priority Level: 1

Class-map: tc6 (match-all)
  0 packets
  Match: traffic-class 6
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000
  Priority: Strict,

  Priority Level: 2

Class-map: tc5 (match-all)
  0 packets
  Match: traffic-class 5
  Priority: Strict,

  Priority Level: 3
  shape (average) cir 2000000000, bc 8000000, be 8000000
  target shape rate 2000000000

Class-map: tc4 (match-all)
  0 packets
  Match: traffic-class 4
  Priority: Strict,

  Priority Level: 4
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000

Class-map: tc3 (match-all)
  0 packets
  Match: traffic-class 3
  Priority: Strict,

  Priority Level: 5
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000
```

```
Class-map: tc2 (match-all)
  0 packets
  Match: traffic-class 2
  Priority: Strict,

  Priority Level: 6
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000

Class-map: tc1 (match-all)
  0 packets
  Match: traffic-class 1
  shape (average) cir 40000000000, bc 400000000, be 400000000
  target shape rate 40000000000
  Priority: Strict,

  Priority Level: 7

Class-map: class-default (match-any)
  35230 packets
  Match: any
  Queueing
  queue limit 7500000 bytes
  (total drops) 0
  (bytes output) 0
  shape (average) cir 100000000, bc 400000, be 400000
  target shape rate 100000000
```

Where to Go Next

Review the auto-QoS documentation to see if you can use these automated capabilities for your QoS configuration.

Additional References for QoS

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	*Command Reference (Ca* *Cisco IOS Quality of Serv*

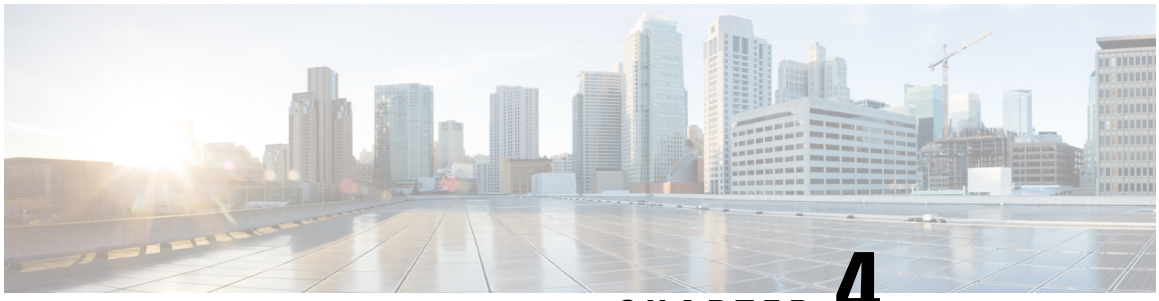
Feature History for QoS

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	QoS Functionality	QoS provides preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to Cisco Feature Navigator.



CHAPTER 4

Configuring Layer 3 Subinterface Queuing

Restrictions for Layer 3 Subinterface Queuing

- Subinterface Queuing is supported only on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).
- Subinterface queuing is supported only on Layer 3 subinterfaces.
- Queuing policy is not supported on port channels or subinterfaces of port channels.
- Subinterface queuing supports a maximum of two queues—priority traffic class (class 7) and nonpriority traffic class (class default).
- Subinterface queuing is not supported for multicast traffic.
- Queued traffic on subinterfaces cannot be re-marked on the main interface using the re-mark policy.
- Hierarchical queuing is supported on the subinterface with this restriction—the parent policy can contain only class-default, which can have shape or bandwidth remaining ratio or both shape and bandwidth remaining ratio.
- Shaping is only supported in:
  - Priority traffic class for child or non-HQoS queuing policy.
  - Class-default for parent policy in HQoS policy.
- Bandwidth remaining ratio is not supported for queue policies with priority level defined.
- Bandwidth remaining ratio is only supported in the parent class of the HQoS queuing policy on subinterfaces.
- Traffic on all the subinterfaces without queuing policy applied, flows through the main interface queues, while traffic for the subinterfaces with queuing policy applied, flows through their respective queues.

- For direct, connected interfaces with defined IP addresses, traffic does not flow through subinterface queues but through the main interface queue. For indirect connected interfaces, traffic flows through subinterface queues.
- The **no switchport** command should be run to configure layer 3 subinterfaces.
- If a policy is applied on any of the subinterfaces, you cannot apply or remove a policy on the main interface without removing the policy on the subinterface. You can only modify the policy on the main interface.
- Queuing policy maps can be applied to a maximum of 400 Layer 3 subinterfaces.
- When HQoS is enabled on a subinterface, and the parent is configured with shape rate value (in percent), then the parent shape rate value is calculated using the main interface physical bandwidth as the reference bandwidth.
- When queuing policy is enabled on a subinterface, the subinterface queuing packet drop statistics are not reflected in the total packet drop statistics at the main interface level.

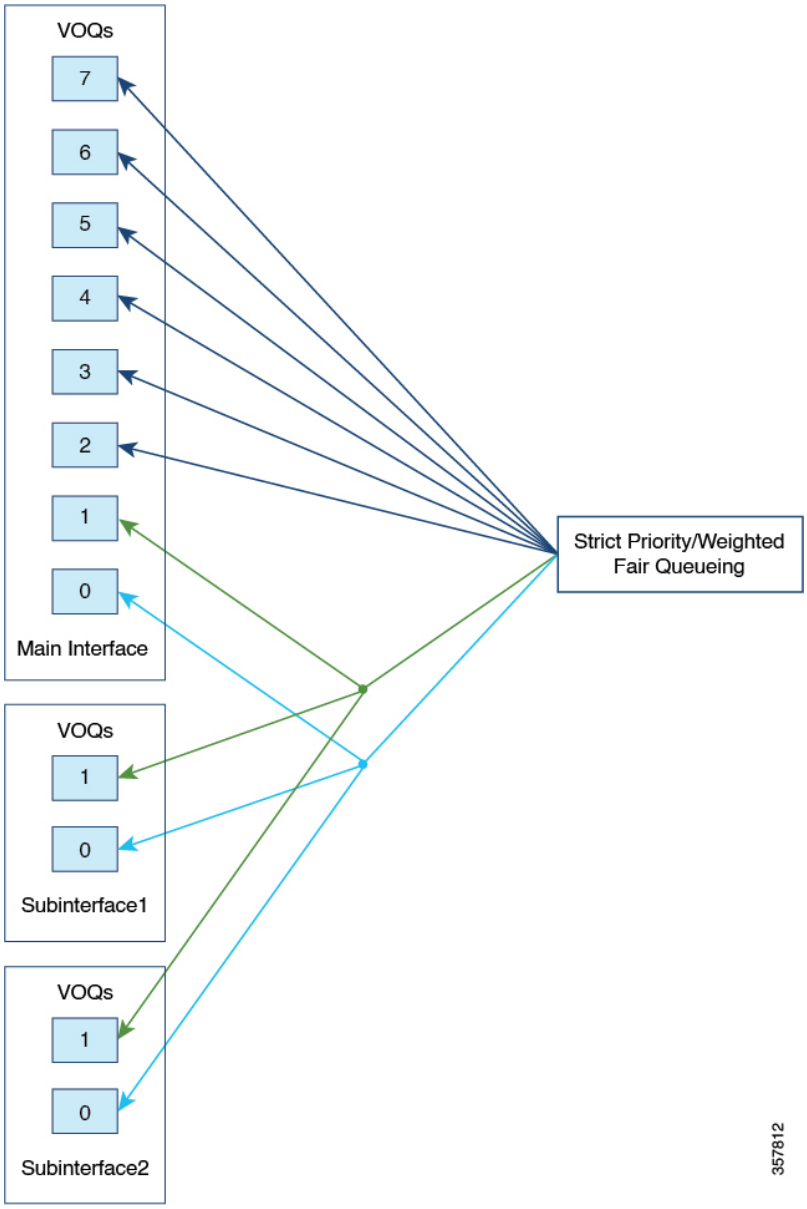
Information About Layer 3 Subinterface Queuing

The following sections describe different queuing modes of layer 3 subinterface.

Default Queuing Mode

In default queuing mode, the main interface queue configuration has absolute priority control over and above the subinterface queues. The main interface has 8 queues (7 to 0 in order) by default, which can be configured in any of [pP(8-p)Q, where p=1 to 7, P=priority, and Q=queue] combination. The subinterface has 2 queues (Q1 and Q0) that map to or share the priority level with queues (Q1 and Q0) of the main interface. Therefore, these subinterface queues are lower in priority when compared to the main interface queues based on the policy applied on the main interface.

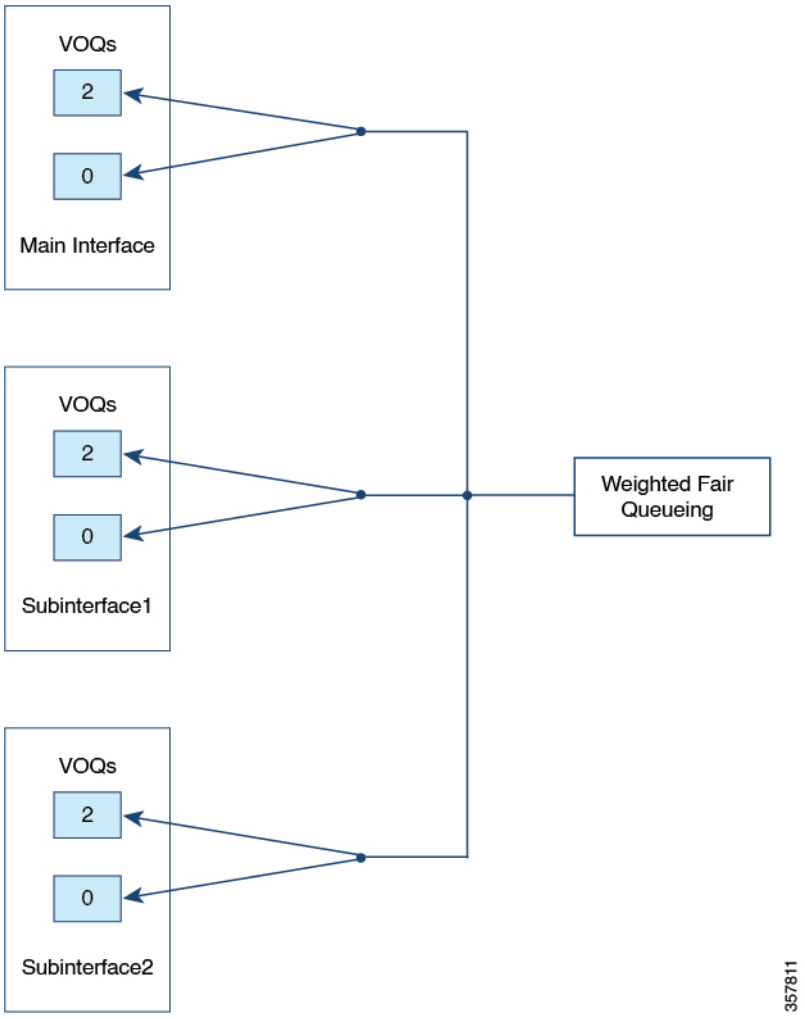
Figure 5: Default Queuing Mode



Subinterface Propagation Queuing Mode

In this mode, the main interface queues are treated and configured just as any other subinterface queue, and all behavior and restriction of the subinterface queues are applicable to the main interface queue as well. A queue of same priority level across the main interface and subinterface(s) contend for bandwidth or priority at the same level. Further, in this mode the user can configure bandwidth distribution ratio between the main interface and subinterface(s). The user cannot change the mode of the main interface while queuing policy is applied on the main interface.

Figure 6: Subinterface Propagation Queuing Mode



Hierarchical QoS

The following HQoS configuration options are supported.

- Port shaper: Port shaper is used to limit or shape the overall transmittion rate for a given port (main or subinterface). Port shaper is supported via HQoS queuing policy on the main interface and subinterface(s) with shaper configured on the parent policy.
  - Main interface: Port shaper applied on the main interface is applicable to all subinterfaces.

Note This is applicable only if the main interface is not in priority propagation mode. If the main interface is in priority propagation mode, then it is treated as a subinterface and its shaper value will not applicable to the subinterfaces.

  - Subinterface: Port shaper is applied only to the subinterface on which port shaper policy is applied.

- Port bandwidth distribution ratio among subinterfaces: Port bandwidth remaining ratio is used to govern or configure bandwidth ditribution ratio between subinterfaces. Port bandwidth remaining ratio is supported using the **bandwidth remaining ratio** command which is configured on the parent policy applied on the subinterface.

  Port bandwidth remaining ratio is also supported between the main interface and subinterface, when subinterface priority propagation mode is enabled.

  A maximum of 7 ratio values are supported, and so even if there are several subinterfaces under a main interface, there can be at-most 7 absolute bandwidth distribution ratio value between them.

Note The value of port shaper set on the main interface will be the maximum allowed value on subinterfaces even if a higher port shaper value is defined on the subinterfaces.

How to Configure Layer 3 Subinterface Queuing

The following sections provide configuration information about layer 3 subinterface queuing.

Enabling Subinterface Queuing Policy

This procedure provides the steps of how to enable subinterface queuing policy.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**class-map** {**match-any** | **match-all**} {*class name*} **Example:** `Device(config)# class-map match-all traffic-class7`	Enters class map configuration mode. • Creates a class map to be used for matching packets to the class whose name you specify. • **match-any**: Any one of the match criteria must be met for traffic entering the traffic class to be classified as part of it.

	Command or Action	Purpose
		• **match-all**: All of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. **Note** **match-all** is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.
Step 4	**match traffic-class** *traffic class value* **Example:** `Device(config-cmap)# match traffic-class 7`	Matches QoS traffic class value (from 1 to 7). **Note** Only traffic-class 7 is supported for queueing on subinterfaces. If it is not a subinterface queue, you can perform steps 3 and 4 to create class maps for all traffic classes (7 to 1), and attach these to queue policy-maps.
Step 5	**exit** **Example:** `Device(config-cmap)# exit`	Exits class map configuration mode and enters global configuration mode.
Step 6	**policy-map type queueing** *policy name* **Example:** `Device(config)# policy-map type queueing subif_q_policy`	Specifies the name of the main interface queueing profile policy and enters policy map configuration mode.
Step 7	**class** *class-name* **Example:** `Device(config-pmap)# class traffic-class7`	Specifies the name of the class to be associated with the policy and enters policy class map configuration mode. Command options for policy class map configuration mode include the following: • *word*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.
Step 8	**shape average** {*Kb/s* | **percent**} **Example:** `Device(config-pmap-c)# shape average percent 10`	Configures the traffic shaping average. The parameters include: • *Kb/s*: Use this command to configure a specific value. The range is 1.2 Mbps to 400 Gbps.

	Command or Action	Purpose
		• **percent**: Allocates a maximum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is divided along all non-priority queues based on bandwidth remaining ratio.
Step 9	**priority level** *level* **Example:** `Device(config-pmap-c)# priority level 1`	Configures multi-level priority queue.
Step 10	**exit** **Example:** `Device(config-pmap-c)# exit`	Exits policy class map configuration mode and enters class map configuration mode.
Step 11	**exit** **Example:** `Device(config-pmap)# exit`	Exits policy map configuration mode and enters global configuration mode.
Step 12	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE1/0/9`	Identifies the main interface and enters interface configuration mode.
Step 13	**no switchport** **Example:** `Device(config-if)# no switchport`	Switches the interface that is in Layer 2 mode into Layer 3 mode for Layer 3 configuration.
Step 14	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 15	**interface** *interface-id.subinterface-id* **Example:** `Device(config)# interface HundredGigE1/0/9.1`	Identifies the subinterface and enters subinterface configuration mode.

	Command or Action	Purpose
Step 16	**encapsulation dot1Q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1Q 11`	Enables IEEE 802.1Q encapsulation of traffic on the subinterface.
Step 17	**service-policy type queueing output** *policy name* **Example:** `Device(config-subif)# service-policy type queueing output subif_q_policy`	Attaches two queue policy-map to the subinterface.
Step 18	**end** **Example:** `Device(config-subif)# end`	Returns to privileged EXEC mode.

Enabling Subinterface Priority Propagation Mode

This procedure provides the steps of enable subinterface priority propagation mode on the main interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE1/0/23`	Identifies the main interface and enters interface configuration mode.
Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Switches the main interface that is in Layer 2 mode into Layer 3 mode for Layer 3 configuration.

	Command or Action	Purpose
Step 5	**queuing mode sub-interface priority-propagation** **Example:** `Device(config-if)# queuing mode sub-interface priority-propagation`	Enables subinterface priority propagation mode. **Note** This mode can be enabled only when no policy is applied on the main interface.
Step 6	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Configuring Hierarchical QoS Policy on Subinterface

This procedure provides the steps to configure HQoS policy on a subinterface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**class-map match-all** {*classname*} **Example:** `Device(config)# class-map match-all traffic-class7`	Configures the class-map to match all criterias for traffic entering the traffic class, and enters class map configuration mode.
Step 4	**match traffic-class** *class value* **Example:** `Device(config-cmap)# match traffic-class 7`	Matches QoS traffic class value.
Step 5	**exit** **Example:** `Device(config-cmap)# exit`	Exits class map configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 6	**policy-map type queueing** *child policy name* **Example:** `Device(config)# policy-map type queueing child`	Specifies the child queueing profile policy and enters policy map configuration mode.
Step 7	**class** *class-name* **Example:** `Device(config-pmap)# class traffic-class7`	Specifies the name of the class to be associated with the policy and enters policy class map configuration mode. Command options for policy class map configuration mode include the following: • *word*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.
Step 8	**shape average** {*Kb/s* | **percent**} **Example:** `Device(config-pmap-c)# shape average 1000000`	Configures the traffic shaping average. The parameters include: • *Kb/s*: Use this command to configure a specific value. The range is 1.2 Mbps to 400 Gbps. • **percent**: Allocates a maximum bandwidth to a particular class.
Step 9	**priority level** *level* **Example:** `Device(config-pmap-c)# priority level 1`	Specifies the priority level of the queue.
Step 10	**exit** **Example:** `Device(config-pmap-c)# exit`	Exits policy class map configuration mode and enters class map configuration mode.
Step 11	**exit** **Example:** `Device(config-pmap)# exit`	Exits policy map configuration mode and enters global configuration mode.
Step 12	**policy-map type queueing** *parent policy name* **Example:** `Device(config)# policy-map type queueing parent`	Specifies the parent queueing profile policy and enters policy map configuration mode.

	Command or Action	Purpose
Step 13	**class class-default** **Example:** `Device(config-pmap)# class class-default`	Specifies the class default to be associated with the parent policy and enters policy class map configuration mode.
Step 14	**shape average** {*port shaper value*} {*Kb/s* | **percent**} **Example:** `Device(config-pmap-c)# shape average 2000000000`	Configures the traffic shaping average.
Step 15	**service-policy** *child policy-map name* **Example:** `Device(config-pmap-c)# service-policy child`	Configures the QoS service policy of the child.
Step 16	**exit** **Example:** `Device(config-pmap-c)# exit`	Exits policy class map configuration mode and enters class map configuration mode.
Step 17	**exit** **Example:** `Device(config-pmap)# exit`	Exits policy map configuration mode and enters global configuration mode.
Step 18	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE1/0/5`	Identifies the interface and enters interface configuration mode.
Step 19	**no switchport** **Example:** `Device(config-if)# no switchport`	Switches the interface that is in Layer 2 mode into Layer 3 mode for Layer 3 configuration.
Step 20	**exit** **Example:** `Device(config-pmap)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 21	**interface** *interface-id.subinterface-id* **Example:**	Identifies the subinterface and enters subinterface configuration mode.

	Command or Action	Purpose
	`Device(config)# interface HundredGigE1/0/5.1`	
Step 22	**encapsulation dot1Q** *vlan-id* **Example:** `Device(config-if)# encapsulation dot1Q 11`	Enables IEEE 802.1Q encapsulation of traffic on the subinterface.
Step 23	**service-policy type queueing output** *parent policy name* **Example:** `Device(config-if)# service-policy type queueing output parent`	Attaches the parent queue policy-map to the subinterface.
Step 24	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuration Examples for Layer 3 Subinterface Queuing

The following sections provide configuration examples for Layer 3 subinterface queuing.

Example: Enabling Subinterface Queuing Policy

The following is an example of how to enable subinterface queuing policy.

```
Device# configure terminal
Device(config)# class-map match-all traffic-class7
Device(config-cmap)# match traffic-class 7
Device(config)# policy-map type queueing llq
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average percent 10
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/9
Device(config-if)# no switchport
Device(config-if)# exit
Device(config)# interface HundredGigE1/0/9.1
Device(config-subif)# encapsulation dot1Q 11
Device(config-subif)# service-policy type queueing output subif_q_policy
Device(config-subif)# end
```

Example: Enabling Subinterface Priority Propagation Mode

The following is an example of how to enable subinterface priority propagation mode.

```
Device# configure terminal
Device(config)# interface HundredGigE1/0/23
Device(config-if)# no switchport
Device(config-if)# queuing mode sub-interface priority-propagation
Device(config-subif)# end
```

Example: Configuring Hierarchical QoS Policy on Subinterface

The following is an example of how to configure hierarchical QoS policy on subinterface.

```
Device# configure terminal
Device(config)# class-map match-all traffic-class7
Device(config-cmap)# match traffic-class 7
Device(config-cmap)# exit
Device(config)# policy-map type queueing child
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average 1000000
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# policy-map type queueing parent
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 2000000000
Device(config-pmap-c)# service-policy child
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/5
Device(config-if)# no switchport
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/5.1
Device(config-if)# encapsulation dot1Q 11
Device(config-if)# service-policy type queueing output parent
Device(config-if)# end
```

The following is an example of how to configure hierarchical QoS bandwidth remaining ratio on subinterface.

```
Device# configure terminal
Device(config)# class-map match-all traffic-class7
Device(config-cmap)# match traffic-class 7
Device(config-cmap)# exit
Device(config)# policy-map type queueing child
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average 1000000
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# policy-map type queueing parent_ratio_5
Device(config-pmap)# class class-default
Device(config-pmap-c)# bandwidth remaining ratio 5
Device(config-pmap-c)# service-policy child
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# policy-map type queueing parent_ratio_10
Device(config-pmap)# class class-default
```

```
Device(config-pmap-c)# bandwidth remaining ratio 10
Device(config-pmap-c)# service-policy child
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/5
Device(config-if)# no switchport
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/5.1
Device(config-if)# encapsulation dot1Q 11
Device(config-if)# service-policy type queueing output parent_ratio_5
Device(config-if)# exit
Device(config)# interface HundredGigE1/0/5.2
Device(config-if)# encapsulation dot1Q 15
Device(config-if)# service-policy type queueing output parent_ratio_10
Device(config-if)# end
```

Monitoring Layer 3 Subinterface Queuing Configuration

The following commands can be used to monitor layer 3 subinterface queuing configuration on the device.

Command	Description
show policy-map type queueing interface *interface-id[.subinterface]*	Displays the runtime representation and statistics of all the queueing policies configured on the device.
show running interface *interface-id[.subinterface]*	Displays the configured interface and values under it.
show running policy-map *name*	Displays a list of the policy maps along with traffic class information.

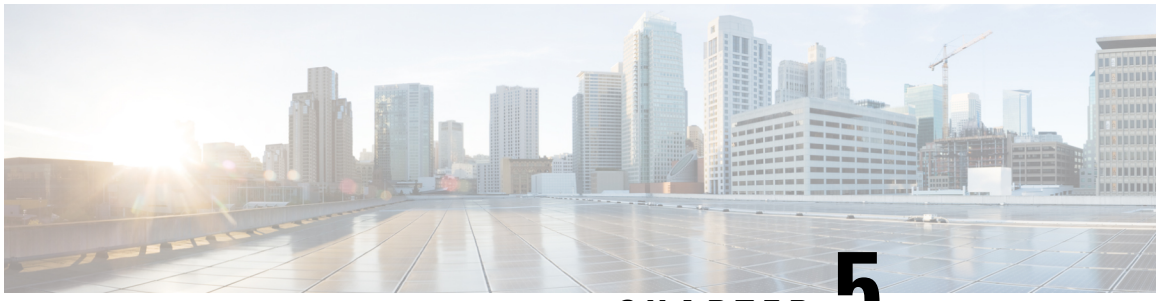
Feature History for Layer 3 Subinterface Queuing

This table provides release and related information for features explained in this module.

These features are available on all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.8.1	Layer 3 Subinterface Queuing	Queuing support on Layer 3 subinterfaces has been introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to Cisco Feature Navigator.



CHAPTER 5

Configuring Weighted Random Early Detection

Avoiding Network Congestion

Heterogeneous networks include different protocols used by applications, giving rise to the need to prioritize traffic in order to satisfy time-critical applications while still addressing the needs of less time-dependent applications, such as file transfer. If your network is designed to support different traffic types that share a single data path between devices in a network, implementing congestion avoidance mechanisms ensures fair treatment across the various traffic types and avoids congestion at common network bottlenecks. Congestion avoidance mechanism is achieved through packet dropping.

Random Early Detection (RED) is a commonly used congestion avoidance mechanism in a network.

Tail Drop

Tail drop treats all traffic equally and does not differentiate within a class of service. When the output queue is full and tail drop is in effect, packets are dropped until the congestion is eliminated and the queue is no longer full.

Weighted Random Early Detection

The RED mechanism takes advantage of the congestion control mechanism of TCP. Packets are randomly dropped prior to periods of high congestion. Assuming the packet source uses TCP, it decreases its transmission

rate until all the packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow down transmission of packets. TCP not only pauses, but also restarts quickly and adapts its transmission rate to the rate that the network can support.

WRED is the Cisco implementation of RED. It combines the capabilities of RED algorithm with IP Precedence or Differentiated Services Code Point (DSCP) or Class of Service (COS) values. On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), WRED combines the capabilities of RED algorithim with discard-class 0 and 1.

How WRED Works

WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion. WRED drops some packets early rather than waiting until the queue is full. Thus it avoids dropping large number of packets at once and minimizes the chances of TCP global synchronization.

Approximate Fair Drop (AFD) is an Active Queue Management (AQM) algorithm that determines the packet drop probability. The probability of dropping packets depends upon the arrival rate calculation of a flow at ingress and the current queue length.

AFD based WRED is implemented on wired network ports.

AFD based WRED emulates the preferential dropping behavior of WRED. This preferential dropping behavior is achieved by changing the weights of AFD sub-classes based on their corresponding WRED drop thresholds. Within a physical queue, traffic with larger weight incurs less drop probability than that of smaller weight.

- Each WRED enabled queue has high and low thresholds.
- A sub-class of higher priority has a larger AFD weight.
- The sub-classes are sorted in ascending order, based on lowest of WRED minThreshold.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), AFD is not used. Drop probability for different queue length and discard-class is programmed in the hardware.

WRED Weight Calculation

AFD weight is calculated using low and high threshold values; AFD is an adjsuted index of the average of WRED high and WRED low threshold values.

When a packet arrives at an interface, the following events occur:

1. The drop probability is calculated. The drop probability increases as the AFD weight decreases. That means, if the average of low and high threshold values is less, the drop probability is more.
2. WRED consideres the priority of packet flows and the threshold values before deciding to drop the packet. The CoS, DSCP or IP Precedence values are mapped to the specified thresholds. Once these thresholds are exceeded, packets with the configured values that are mapped to these thresholds are eligible to be dropped. Other packets with CoS, DSCP or IP Precedence values assigned to the higher thresholds are en-queued. This process keeps the higher priority flows intact and minimises the latency in packet transmission.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), discard class 0 and 1 are mapped to the specified thresholds.

3. If packets are not dropped using WRED, they are tail-dropped.

Limitations for WRED Configuration

- Weighted Tail Drop (WTD) is enabled by default on all the queues.
- WRED can be enabled / disabled per queue. When WRED is disabled, WTD is adapted on the target queue. Policy-map with WRED profile is configured only on physical ports as output policy.
- WRED is supported only in network port queues and is not supported on internal CPU queues and stack queues.
- Each WRED physical queue can support three different threshold pairs. Each pair is for one QoS tag value.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), WRED threshold pair configuration are in percentage of queue-limit for that particular queue. WRED can be configured for seven queues in the queueing policy, and upto 2 WRED threshold pairs per queue are supported. One threshold pair is for discard-class 0 and the other for discard-class 1.

- WRED based on DSCP, PREC, or COS are not supported on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2). WRED is only based on discard-class.
- Ensure that you configure bandwidth or shape in the policy-map along with WRED.

Note On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), ensure that you configure bandwidth remaining ratio or shape in the policy-map along with WRED.

- Specify all the WRED thresholds only in percentage mode.
- Map the WRED threshold pairs by mapping class-map filter with corresponding match filters.

  We recommend the class-map with match “any” filter.
- WRED for priority traffic is not supported.
- WRED and queue limit are not supported for the same policy.

Note WRED and queue limit for the same policy are supported on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

- Wired ports support a maximum of eight physical queues, of which you can configure WRED only on four physical queues, each with three threshold pairs. The remaining queues are configured with WTD. Policies with more than four WRED queues are rejected.

Note This restriction is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Usage Guidelines for WRED

To configure AFD based WRED feature, specify the policy map and add the class. Use the **random-detect** command to specify the method (using the dscp-based / cos-based / precedence-based /discard-class-based arguments) that you want WRED to use to calculate the drop probability.

Note You can modify the policy on the fly. The AFD weights are automatically recalculated.

WRED can be configured for any kind of traffic like IPv4/IPv6, Multicast, and so on. WRED is supported on all 8 queueing classes.

On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), WRED configuration is based only on discard-class, and WRED threshold pair configuration are in percentage of queue-limit for that particular queue. By configuring threshold values for discard-class 0 and 1 separately, WRED for that particular queue is achieved. Similarly, this can be done for all the seven queues in the queueing policy.

Consider the following points when you are configuring WRED with **random-detect** command:

- With dscp-based argument, WRED uses the DSCP value to calculate drop probability.
- With cos-based argument, WRED uses the COS value to calculate drop probability.
- With discard-class-based argument, WRED uses the discard-class value to calculate drop probability.
- By default, WRED uses the IP Precedence value to calculate drop probability. **precedence-based** argument is the default and it is not displayed in the CLI.

Note **show run policy-map** *policy-map* command does not display “precedence” though precedence is configured with **random-detect** command.

- The dscp-based and precedence-based arguments are mutually exclusive.
- Each of the eight physical queues can be configured with different WRED profiles.

Configuring WRED

Configuring WRED based on DSCP Values

Use the following steps to configure WRED profile based on DSCP values in packet mode:

Note This procedure is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Procedure

	Command or Action	Purpose
Step 1	**class-map** *match-criteria class-name* **Example:** `device(config)# class-map match-any CS`	Configures match criteria for the class map. Recommended match-criteria is match-any
Step 2	**match** *class-map-name* **Example:** `device(config-cmap)#match dscp CS1`	Match a class-map.
Step 3	**policy-map** *name* **Example:** `device(config)#policy-map PWRED`	Specifies the name of the WRED profile policy to be created.
Step 4	**class** *class-name* **Example:** `device(config-pmap)#class CS`	Specifies the name of the Class to be associated with the policy.
Step 5	Use either **bandwidth** {*kbps* | **remaining ratio** | **percent** *percentage*} or **shape** { **average** | **peak** } *cir* **Example:** `device(config-pmap-c)#bandwidth percent 10`	Specify either the bandwidth allocated for a class belonging to a policy map or the traffic shaping.
Step 6	**random-detect** *dscp-based* **Example:** `device(config-pmap-c)#random-detect dscp-based`	Configures WRED to use the DSCP value when it calculates the drop probability for the packet.
Step 7	**random-detect dscp** *dscp-value* **percent** *minThreshold maxThreshold* **Example:**	Specifies the minimum and maximum thresholds, in percentage.

	Command or Action	Purpose
	`device(config-pmap-c)#random-detect dscp cs1 percent 10 20`	
Step 8	**interface** *interface-name* **Example:** `device(config)#interface HundredGigE1/0/2`	Enters the interface configuration mode.
Step 9	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Configuring WRED based on Class of Service Values

Use the following steps to configure WRED profile based on Class of Service (COS) values in packet mode:

Note This procedure is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Procedure

	Command or Action	Purpose
Step 1	**class-map** *match-criteria class-name* **Example:** `device(config)# class-map match-any CS`	Configures match criteria for the class map. Recommended match-criteria is match-any
Step 2	**match** *class-map-name* **Example:** `device(config-cmap)#match cos 3`	Match a class-map.
Step 3	**policy-map** *name* **Example:** `device(config)#policy-map PWRED`	Specifies the name of the WRED profile policy to be created.
Step 4	**class** *class-name* **Example:** `device(config-pmap)#class CS`	Specifies the name of the Class to be associated with the policy.
Step 5	**bandwidth** {*kbps*| **remaining** *percentage* | **percent** *percentage*} **Example:** `device(config-pmap-c)#bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map.

	Command or Action	Purpose
Step 6	**random-detect** *cos-based* **Example:** `device(config-pmap-c)#random-detect cos-based`	Configures WRED to use the CoS value when it calculates the drop probability for the packet.
Step 7	**random-detect cos** *cos-value* **percent** *minThreshold maxThreshold* **Example:** `device(config-pmap-c)#random-detect cos 3 percent 10 20`	Specifies the minimum and maximum thresholds, in percentage.
Step 8	**interface** *interface-name* **Example:** `device(config)# interface HundredGigE1/0/2`	Enters the interface configuration mode.
Step 9	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Configuring WRED based on IP Precedence Values

Use the following steps to configure WRED profile based on IP precedence values in packet mode:

Note This procedure is not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Procedure

	Command or Action	Purpose
Step 1	**class-map** *match-criteria class-name* **Example:** `device(config)# class-map match-any CS`	Configures match criteria for the class map. Recommended match-criteria is match-any
Step 2	**match** *class-map-name* **Example:** `device(config-cmap)#match precedence 3`	Match a class-map.
Step 3	**policy-map** *name* **Example:** `device(config)#policy-map pwred`	Specifies the name of the WRED profile policy to be created.

	Command or Action	Purpose
Step 4	**class** *class-name* **Example:** `device(config-pmap)#class CS`	Specifies the name of the Class to be associated with the policy.
Step 5	**bandwidth** {*kbps*| **remaining** *percentage* | **percent** *percentage*} **Example:** `device(config-pmap-c)#bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 6	**random-detect** *precedence-based* **Example:** `device(config-pmap-c)#random-detect precedence-based`	Configures WRED to use the IP precedence value when it calculates the drop probability for the packet.
Step 7	**random-detect precedence** *precedence-value* **percent** *minThreshold maxThreshold* **Example:** `device(config-pmap-c)#random-detect precedence 3 percent 10 20`	Specifies the minimum and maximum thresholds, in percentage.
Step 8	**interface** *interface-name* **Example:** `device(config)#interface HundredGigE1/0/2`	Enters the interface configuration mode.
Step 9	**service-policy output** *policy-map* **Example:** `device(config-if)#service-policy output pwred`	Attaches the policy map to an output interface.

Configuring WRED based on Discard-class Values

Use the following steps to configure WRED profile based on discard-class values in packet mode:

Note This configuration procedure is applicable only to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP-2).

Procedure

	Command or Action	Purpose
Step 1	**policy-map type queueing** *name* **Example:** `Device(config)# policy-map queueing wred`	Specifies the name of the WRED queueing profile policy to be created.

	Command or Action	Purpose
Step 2	**class** *class-name* **Example:** `Device(config-pmap)# class tc2`	Specifies the name of the class to be associated with the policy.
Step 3	**shape average percent** *percentage* **Example:** `Device(config-pmap-c)# shape average percent 10`	Specifies the traffic shaping percentage bandwidth for committed information rate.
Step 4	**random-detect discard-class-based** **Example:** `Device(config-pmap-c)# random-detect discard-class-based`	Configures WRED to use the discard-class value when it calculates the drop probability for the packet.
Step 5	**random-detect discard-class** *value* **percent** *minThreshold maxThreshold mark-probability-denominator* **Example:** `Device(config-pmap-c)# random-detect discard-class 0 percent 20 80 1`	Configures WRED discard-class 0, and specifies the minimum and maximum thresholds, in percentage.
Step 6	**random-detect discard-class** *value* **percent** *minThreshold maxThreshold mark-probability-denominator* **Example:** `Device(config-pmap-c)# random-detect discard-class 1 percent 15 70 1`	Configures WRED discard-class 1, and specifies the minimum and maximum thresholds, in percentage.
Step 7	**end** **Example:** `Device(config-pmap-c)# end`	Exits WRED class action configuration mode and returns to privileged EXEC mode.

WRED Configuration Example

The following example enables WRED to use DSCP profile for class CS. It configures three sub-classes cs1, cs2, and cs3 with their WRED minimum and maximum thresholds and finally applies the policy to Hundred Gigabit Ethernet interface 8:

```
Device(config)# class-map match-any CS
Device(config-cmap)# match dscp cs1
Device(config-cmap)# match dscp cs2
Device(config-cmap)# match dscp cs3
Device(config-cmap)# policy-map PWRED
Device(config-pmap)# class CS
Device(config-pmap-c)# bandwidth ratio 10
Device(config-pmap-c)# random-detect dscp-based
Device(config-pmap-c)# random-detect dscp cs1 percent 10 20
```

```
Device(config-pmap-c)# random-detect dscp cs2 percent 20 30
Device(config-pmap-c)# random-detect dscp cs3 percent 34 44
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/8
Device(config-if)# service-policy output PWRED
```

The following is a WRED configuration example of the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

```
Device# configure terminal
Device(config)# policy-map type queueing wred
Device(config-pmap)# class tc7
Device(config-pmap-c)# shape average percent 40
Device(config-pmap-c)# class tc6
Device(config-pmap-c)# shape average percent 20
Device(config-pmap-c)# random-detect discard-class-based
Device(config-pmap-c)# random-detect discard-class 0 percent 20 80 1
Device(config-pmap-c)# random-detect discard-class 1 percent 15 70 1
```

WRED Support with Hierarchical QoS

Hierarchical QoS allows you to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management.

For HQoS, WRED is allowed only on the child policy and not on the parent policy. You can have the shaping configured on the parent policy and WRED on the child.

The following example configures the parent policy **pwred-parent** with traffic shaped on the basis of 10 percent of the bandwidth, that applies to its child, **pwred-child** configured for DSCP-based WRED.

```
policy-map PWRED-CHILD
 class CWRED
    bandwidth percent 10
    random-detect dscp-based
    random-detect dscp 1 percent 10 20
    random-detect dscp 10 percent 20 30

policy-map PWRED-PARENT
  class class-default
  shape average percent 10
  service-policy PWRED-CHILD
```

The following are WRED and HQoS WRED sample configuration for the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

```
Device(config)# policy-map type queueing wred
Device(config-pmap)# class tc7
Device(config-pmap-c)# shape average percent 40
Device(config-pmap-c)# class tc6
Device(config-pmap-c)# shape average percent 20
Device(config-pmap-c)# random-detect discard-class-based
Device(config-pmap-c)# random-detect discard-class 0 percent 20 80 1
Device(config-pmap-c)# random-detect discard-class 1 percent 15 70 1

Device(config)# policy-map type queueing hqos_wred
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average percent 80
```

```
Device(config-pmap-c)# service-policy wred
Device(config-pmap-c)# end
```

The following show commands for the HQoS WRED and WRED configuration is for the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

```
Device# show policy-map type queueing hqos_wred

Policy Map type queueing hqos_wred
Class class-default
Average Rate Traffic Shaping
cir 80%
service-policy wred

Device# show policy-map type queueing wred
Policy Map type queueing wred
Class tc7
Average Rate Traffic Shaping
cir 40%
Class tc6
Average Rate Traffic Shaping
cir 20%
percent-based wred, exponential weight 1

discard-class min-threshold max-threshold mark-probablity
------------------------------------------------------------
0 20 80 1/1
1 15 70 1/1
2 - - 1/1
3 - - 1/1
4 - - 1/1
5 - - 1/1
6 - - 1/1
7 - - 1/1
```

Displaying WRED Configuration

The following example shows how to display the WRED and threshold labels:

```
Device# show policy-map PWRED

  Policy Map PWRED
    Class CS
      bandwidth 10 (%)
       percent-based wred

      dscp    min-threshold    max-threshold
      ------------------------------------
      cs1 (8)      10               20
      cs2 (16)     20               30
      cs3 (24)     34               44
      default (0)   -
```

The following example shows how to display WRED AFD Weights, WRED Enq (in Packets and Bytes), WRED Drops (in Packets and Bytes), Configured DSCP labels against the Threshold pairs:

Note Use this command only after you initiate the traffic. **show policy-map interface** is updated with WRED configuration only after a traffic is sent.

```
Device# show policy-map interface HundredGigE 1/0/2

HundredGigE1/0/2

  Service-policy output: PWRED

    Class-map: CS (match-any)
      0 packets
      Match:  dscp cs1 (8)
      Match:  dscp cs2 (16)
      Match:  dscp cs3 (24)
      Queueing

      (total drops) 27374016
      (bytes output) 33459200081
      bandwidth 10% (1000000 kbps)

AFD WRED STATS BEGIN
Virtual Class   min/max        Transmit                    Random drop                 AFD
 Weight


   0          10 / 20         (Byte)33459183360           27374016                    12
                              (Pkts)522799759                427716
              dscp : 8


   1          20 / 30           Byte)0                      0                         20
                                 (Pkts)0                     0
              dscp : 16


   2          34 / 44         (Byte)16721                   0                         31
                                (Pkts)59                    0
              dscp : 24

     Total Drops(Bytes)   : 27374016

     Total Drops(Packets) : 427716
AFD WRED STATS END


    Class-map: class-default (match-any)
      0 packets
      Match: any


      (total drops) 0
      (bytes output) 192
```

Displaying WRED Configurations of the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2)

The following is a sample output from the **show policy-map** command:

```
Device# show policy-map WRED

Policy Map type queueing wred
Class tc7
Average Rate Traffic Shaping
cir 10%
percent-based wred, exponential weight 1

discard-class min-threshold max-threshold mark-probablity
```

```
------------------------------------------------------------
0 20 80 1/1
1 15 70 1/1
2 - - 1/1
3 - - 1/1
4 - - 1/1
5 - - 1/1
6 - - 1/1
7 - - 1/1
```

The following is a sample output of the **show policy-map type queueing interface** command:

```
Device(config)# interface TwentyFiveGigE6/0/1
Device(config-if)# service-policy type queueing output wred
Device(config-if)# end

Device# show policy-map type queueing interface TwentyFiveGigE6/0/1

TwentyFiveGigE6/0/1

  Service-policy queueing output: wred

    Class-map: tc7 (match-all)
      0 packets
      Match: traffic-class 7
      Queueing
      queue limit 30000000 bytes
      (total drops) 0
      (bytes output) 0
      shape (average) cir 20000000000, bc 200000000, be 200000000
      target shape rate 20000000000

    Class-map: tc6 (match-all)
      0 packets
      Match: traffic-class 6
      Queueing
      queue limit 7500000 bytes
      (total drops) 0
      (bytes output) 0
      shape (average) cir 10000000000, bc 100000000, be 100000000
      target shape rate 10000000000
        Exp-weight-constant: 1 (1/2)
        Mean queue depth: 0
            Minimum Maximum  Mark
                    thresh  thresh  prob
        0              20      80  1/1
        1              15      70  1/1
        2               0       0  1/1
        3               0       0  1/1
        4               0       0  1/1
        5               0       0  1/1
        6               0       0  1/1
        7               0       0  1/1

    Class-map: class-default (match-any)
      0 packets
      Match: any

      queue limit 75000000 bytes
      (total drops) 0
      (bytes output) 0
```

Best Practices for WRED Configuration

Note All best practices listed below are not applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), except for **Discard-class Based Configuration**.

- **Support for three WRED configuration pairs**

  Each WRED Physical Queue (AFD Queue) can support three WRED configuation pairs, with unique WRED threshold pair configuration.

```
Policy-map P1
    Class CS
        Random-detect dscp-based
        Random-detect dscp CS1 percent 10 20      // WRED pair 1
        Random-detect dscp CS2  percent 20 30      // WRED pair 2
        Random-detect dscp CS3 percent 30 40      // WRED pair 3
Class-map match-any CS
        match cs1
        match cs2
        match cs3
```

- **Appending WRED configuration pairs**

  You can add overlapping threshold pairs into the WRED configuration pairs.

```
Policy-map P1
    Class CS
       Random-detect dscp-based
       Random-detect dscp CS1 percent 10 20       // WRED pair 1
       Random-detect dscp CS2 percent 20 30       // WRED pair 2
       Random-detect dscp CS3 percent 30 40       // WRED pair 3
       Random-detect dscp CS4 percent  30 40     ==> belongs to WRED pair 3
       Random-detect dscp CS5 percent 20 30     ==> belongs to WRED pair 2
Class-map match-any CS
       match cs1
       match cs2
       match cs3
       match cs4  >>
       match cs5  >>
```

- **Default WRED pairs**

  If less than three WRED pairs are configured, any class-map filter participating WRED gets assigned to the third default WRED pair with maximum threshold (100, 100).

```
Policy-map P1
    Class CS
        Random-detect dscp-based
        Random-detect dscp CS1 percent 10 20   // WRED pair 1
        Random-detect dscp CS2 percent 20 30   // WRED pair 2
Class-map match-any CS
        match CS1
        match CS2
        match CS3
        match CS4
```

  In this case, classes CS3 and CS4 are mapped to WRED pair 3 with threshold (100, 100).

- **Rejection of Mismatched Configuration**

If you configure random-detect without matching filters in a class-map, the policy installation is rejected.

```
Class-map match-any CS
       match CS1
       match CS2
       match CS5
Policy-map P1
    Class CS
        Shape average percent 10
        Random-detect dscp-based
        Random-detect dscp CS1 percent 10 20   // WRED pair 1
        Random-detect dscp CS2 percent 20 30   // WRED pair 2
       Random-detect dscp CS3 percent 30 40   // WRED pair 3  ===> Mismatched sub-class.
```

When this policy is applied to the interface on the egress side, the policy fails during installation as the class-map values are incorrect:

```
device(config)# int Fo1/0/5
device(config-if)# service-policy output P1
device(config-if)#
*Feb 20 17:33:16.964: %IOSXE-5-PLATFORM: Switch 1 R0/0: fed: WRED POLICY INSTALL
FAILURE.Invalid WRED filter mark: 24 in class-map: CS
*Feb 20 17:33:16.965: %FED_QOS_ERRMSG-3-LABEL_2_QUEUE_MAPPING_HW_ERROR: Switch 1 R0/0:
 fed: Failed to detach queue-map for FortyGigabitEthernet1/0/5: code 2
```

- **Discard-class Based Configuration**

  On the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), the minimum and maximum threshold values of the random-detect discard-class 1 must be less than or equal to that of the random-detect discard-class 0.

```
policy-map type queueing wred
class tc7
shape average percent 10
random-detect discard-class-based
random-detect discard-class 0 percent 20 80 1
random-detect discard-class 1 percent 15 70 1
```

Feature History for WRED

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Weighted Random Early Detection mechanism	WRED is a mechanism to avoid congestion in networks. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion, thus avoiding large number of packet drops at once. You can configure WRED to act based on any of the following values: • Differentiated Service Code Point • IP Precedence • Class of Service
Cisco IOS XE Cupertino 17.7.1	Weighted Random Early Detection mechanism	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to Cisco Feature Navigator.